НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи та методи штучного інтелекту»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Інформаційно-аналітична система для визначення типів горних порід за даними геофізичних досліджень свердловин»

Виконав:
студент IV курсу, групи КА-66
Венгер Володимир Олегович _____

Керівник:
асистент кафедри ММСА
Макуха Михайло Павлович _____

Консультант з економічного роділу:
доцент, к.е.н., Шевчук Олена Анатоліївна _____

Консультант з нормоконтролю:
доцент, к.т.н. Коваленко А.Є. _____

Рецензент:
_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інтелектуальний аналіз даних в управлінні проектами»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Оксана ТИМОЩУК

«25» травня 2020 р.

ЗАВДАННЯ

на дипломну роботу студенту

Венгеру Володимиру Олеговичу

1. Тема роботи «Статистична модель прогнозування вартості автомобіля за даними автомобільного ринку України», керівник роботи Макуха Михайло Павлович, асистент кафедри ММСА, затверджені наказом по університету від «25» травня 2020 р. № 1143-с

2. Термін подання студентом роботи 8.06.2020.

3. Вихідні дані до роботи: дані літологіних досліджень свердловини. Взяті у нафтосервісній компанії Schlumberger.

4. Зміст роботи: аналітичний огляд даних геофізичних досліджень, теоретичні основи кластеризаційного аналізу, найбільш застосовувані методи кластеризації та їх модифікації, ідентивікація корисних копалин на основі даних геофізичних досліджень, функціонально-вартісний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) кластеризація, петрофізика, літологія, алгоритм DBSCAN, алгоритм OPTICS, алгоритм TSNE, каротажі.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук О.А., доцент		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Формулювання тематики (напряму) дослідження.	03.06.2019 – 30.09.2019	
2	Аналіз актуальності задач стосовно тематики дослідження	01.10.2019 – 30.10.2019	
3	Аналіз відомих результатів стосовно тематики дослідження	01.11.2019 – 30.11.2019	
4	Формулювання задач дослідження	01.12.2019 – 30.12.2019	
5	Уточнення теми дипломної роботи	25.02.2019	
6	Збір статичних даних, попередній аналіз даних	01.03.2020 – 30.03.2020	
7	Розробка програмного продукту для виконання обчислювальних експериментів	01.03.2020 – 30.04.2020	
8	Виконання обчислювальних експериментів, аналіз та оформлення результатів	01.05.2020 – 20.05.2020	
9	Оформлення пояснювальної записки у цілому	21.05.2020 – 31.05.2020	
10	Підготовка презентації для захисту	28.05.2020 – 01.06.2020	
11	Попередній захист дипломної роботи	01.06.2020 – 03.06.2020	
12	Захист дипломної роботи	15.06.2020 – 18.06.2020	

Студент _____ Володимир Венгер

_____

Керівник _____ Михайло Макуха

РЕФЕРАТ

Дипломна робота містить: 108 с., 9 табл., 21 рис., 2 дод. та 19 джерел. МЕТОДИ КЛАСТЕРИЗАЦІЇ, СВЕРДЛОВИНИ, КЛАСТЕРИ, КОРИСНІ КОПАЛИНИ, КАРОТАЖІ, ОЦІНКИ КЛАСТЕРИЗАЦІЇ.

Метою роботи є визначення найкращої методу для виконання кластеризації корисної копалини використовуючи дані нафтодобувної компанії Schlumberger.

В роботі проведено дослідження застосування різноманітних методів кластеризації та виявлення найкращих до цієї задачі на основі існуючих даних 2020 року. Виділено основні властивослі моделей, які впливають на якість кластеризаці.

У ході дослідження було встановлено, що методи на основі щільності вибірки, такі як DBSCAN і OPTICS, дали хороші результати на досліджуваних даних. Планується розвивати роботу у напрямку дослідження застосування данних методів з метою подальшого зменшення похибки кластеризаці на різних результатах гео-досліджень свердловин.

ABSTRACT

The bachelors work contains of: 108 p., 9 tables, 21 fig., 2 add. and 19 references.

CLUSTERING METHODS, WELLSM CLUSTERS, MINERALS, LOGS, CLUSTERING SCORES.

The aim of the work is to determine the best model for making cluster analyse by using well logging data from Schlumberger oil-company.

The study of the application of using various clustering methods and determine the most appropriate for this dataset according to existence knowledge of 2020 year. Extracted main model properties which influence on clustering result quality.

In course of the study, it was found that the method based on dataset density characteristics, such as DBSCAN and OPTICS, return good results on input data. It is planned to develop work in the direction of research of using this methods in aim further decreasing clustering error rate on different well loging researches.

ЗМІСТ

РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ **12**

1.1 Основі визначення **12**

1.2 Гамма каротаж (GR) **13**

1.3 Спонтанний потенціал (SP) **16**

1.4 Діаметр свердловини (CALI) **17**

1.5 Щільність пласта (DEN) **19**

1.6 Фотоелектричне поглинання (PE) **21**

1.7 Нейронна пористість (NEUT) **22**

1.8 Питомий опір (RT) **26**

1.9 Акустика (DT) **27**

1.10 Висновки до розділу 1 ..... **28**

РОЗДІЛ 2 МАТЕМАТИЧНЕ ОБҐРУНТУВАННЯ ВИКОРИСТОВУВАНИХ МЕТОДІВ КЛАСТЕРИЗАЦІЇ **30**

2.1 Вступ до розділу 2 ..... **30**

2.2 Алгоритм K-Means **31**

2.3 Алгоритм Affinity Propagation **32**

2.4 Алгоритм Spectral Clustering **34**

2.5 Алгоритм Spectral Biclustering **36**

2.6 Алгоритм Agglomerative Clustering **38**

2.7 Алгоритм DBSCAN **40**

2.8 Алгоритм OPTICS **42**

2.9 Оцінки методів кластеризації **43**

2.10 Постановка задачі багатокритеріальної оптимізації. Оптимальність по Парето. **46**

2.11 Метод зменшення розмірності TSNE. **48**

2.12 Висновки до розділу 2 ..... **50**

РОЗДІЛ 3 РОЗРОБКА КОМП'ЮТЕРНОЇ ПРОГРАМИ ДЛЯ АНАЛІЗУ ДАНИХ **51**

3.1 Використані програмні засоби **51**

3.2 Вхідні данні та їх аналіз **52**

3.3 Нормалізація і кластеризація **57**

3.4 Оцінка результатів кластеризацій **58**

3.5 Результати кластеризацій. **61**

3.6 Висновки до розділу 3 ..... **69**

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ 70

4.1 Вступна частина 70

4.3 Виділення основних функційаратетрів ПП 71

4.4 Обґрунтуваннясистемипараметрівпрограмного продукту 74

4.5 Економічний аналіз варіантів розробки 78

4.6 Висновки. Вибіркращоговаріанта ПП техніко-економічногорівня. 83

ВИСНОВКИ ПО РОБОТІ ТА РЕКОМЕНДАЦІЇ ЩОДО ПОДАЛЬШИХ ДОСЛІДЖЕНЬ 84

ЛІТЕРАТУРА 85

ДОДАТОК А ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ 87

ДОДАТОК Б ТЕКСТ ПРОГРАМИ 98

ВСТУП

В великих компаніях, що спеціалізуються на добутку корисних копалин, а саме: залізо, граніт, будівельний пісок, глина, сіль, гравій, гіпс, нафта, газ, золото тощо - аналіз ґрунтів проводиться методом «замірів високої чіткості». Його ідея полягає у бурінні землі на декілька кілометрів вниз, за допомогою підземних зондів, які оснащені різноманітними високочутливими сенсорами. Результати їх дії та принцип роботи розглянемо далі. Такі заміри беруться на широкій області поверхні землі в багатьох точках – на площі в 100 метрів, зазвичай, проводять заміри в радіусі приблизно в 30 метрів, так щоб покрити всю область [15]. А заміри іноді проводяться на областях в декілька квадратних кілометрах – це є великий об’єм даних – матриця m на n (де m – кількість сенсорів різного застосування, n – відстань на яку було проведене буріння, заміри беруться кожні 10 сантиметрів – 0.1 метр) помножений на кількість замірів як показано на рисунку 0.1.

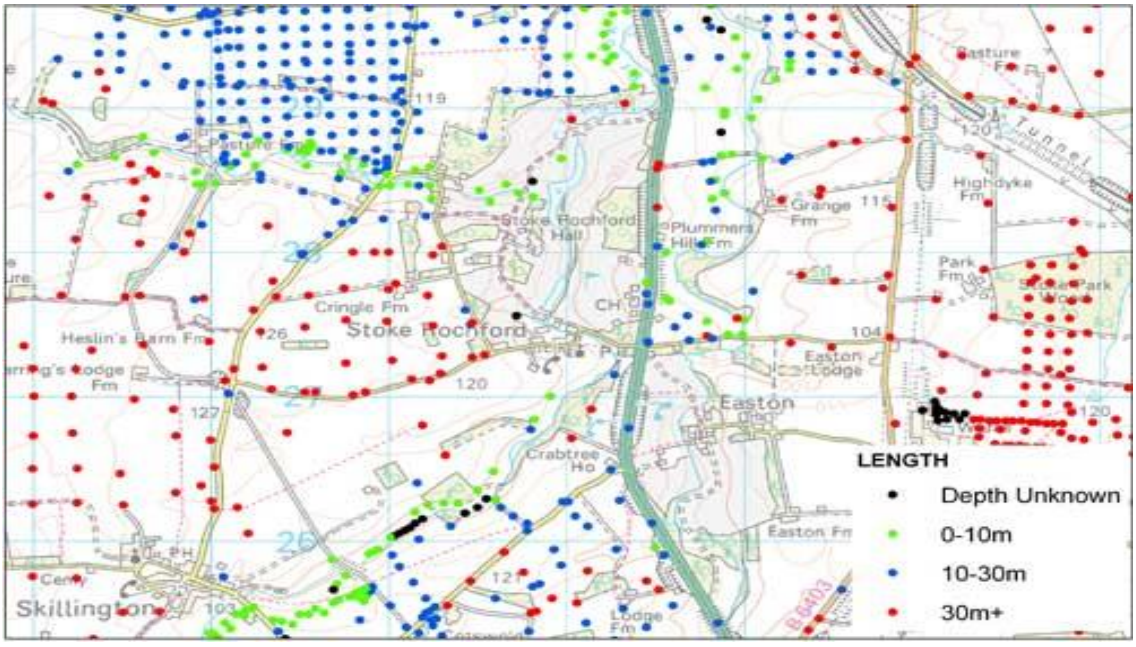


Рисунок 0.1 - Карта розміщення застосування свердловин

Тому питання проведення поверхневого швидкго аналізу великих об’ємів вхідних даних є дуже актуальним для цих галузей. Саме тому в цій роботі будуть розглядатися різні методи кластеризації їх результати і через те що це є задачею навчання моделі без вчителя, тобто надходять ніяк не розмічені дані.

Можна було б також розглянути варіант побудови нейронної мережі яка б попередньо була б дуже добре навчена якраз завдяки великим наборам даних, але цей варіант відпадає через те, що оснащення – свердловини і зонди іноді мають різни габаритні характеристики та кожен раз, під час монтування (підготовки техніки до буріння) налаштування багатьох сенсорів збивається і в наслідок чого перекалібруються, іноді навіть в ручну, а також погодні які змінюються і в навіть дощ, що пройшов тиждень тому – вода осідає в ґрунти і це вже інше налаштування приладів. Всі ці фактори унеможливлюють використання нейронних мереж, адже навчання на даних на таких «динамічних» даних призведе до не коректного результату аналізу нейронною мережею.

Для цієї роботи було надано справжній результат сканування однієї з таких свердловин, але, нажаль результат лише одної і не було ніяких даних щодо типу устаткування, його виробника та налаштування, тому якість і результати кластеризації будуть залежати від якості заміру лише одного такого сканування, і можуть бути інтерпретовані спеціалістом предметної області. В чому даний програмний йому – експерту, допоможе зробити шляхом надання опрацьованих результатів в вигляді різноманітних графіків і матриць, для легкого і швидкого проведення поверхневої оцінки результатів заміру всіх свердловин.

Кластерний аналіз або кластеризація – об'єднання об'єктів вхідних даних в групи (кластери) таким чином, щоб в кожному кластері об’єкт якомога менше відрізняв від усередненої, для конкретного кластера, точки – центр кластера, координати якої – динамічне значення, незмінного, для рідного набору даних, списку характеристик. Це і є метою виявлення даних, з

метою їх узагальнення для зменшення розмірності вхідного набору даних. Це дуже корисно для однорідних великих об’ємів даних, які присутні у багатьох областях, включаючи машинне навчання, розпізнавання корисних копалин гірської та нафтових промисловостях, аналіз літології, пошук інформації та стиснення даних. Завдання кластеризації відноситься до задач статистичної обробки, а також до класу завдань навчання без вчителя.

Важливим, іноді неоднозначним, тому і складним, щодо результатів, являється етап комплексних геофізичних досліджень, та інтерпретація результатів: геологічний; гідрологічний; інженерно-геологічний; мерзлотно-гляціологічний; екологічний. Всі ці дослідження відбуваються за допомогою науки – петрофізика, та її досліджень.

Петрофізика(фізика гірських порід) – це прикладний розділ науки про Землю, який знаходиться на перехресті геології, геофізики, а також фізичних досліджень Землі і фізики речовин[16]. Петрофізика досліджує різноманітні фізичні характеристики гірських порід, кореляцію між ними, кореляцію з фізичними полями Землі, геолого-гідрологічний та пертофізичний взаємозв’язок між їх характеристиками. Дослідження, зазвичай, проводяться на записах – каротажних журналах як показано на рисунку 0.2, спеціальних сенсорів, під час буріння свердловиною, яка аналізує породу на наявність корисних копалин. Свердловина може працювати на повноводних місцевостях – океан, море, річка і тд., від цього також буде змінюватись налаштування і тип устаткування та сенсорів.

Мета роботи – розробити алгоритм, який допоможе класифікувати корисні копалини, на тому чи іншому рівні під землею, з допомогою найбільш відомих методів кластеризації, реалізація яких присутня в бібліотеці мови програмування Python - scikit-learn, та інтерпретація даних у більш придатному для людського сприйняття вигляді, з допомогою методу зниження розмірності даних та графіків.

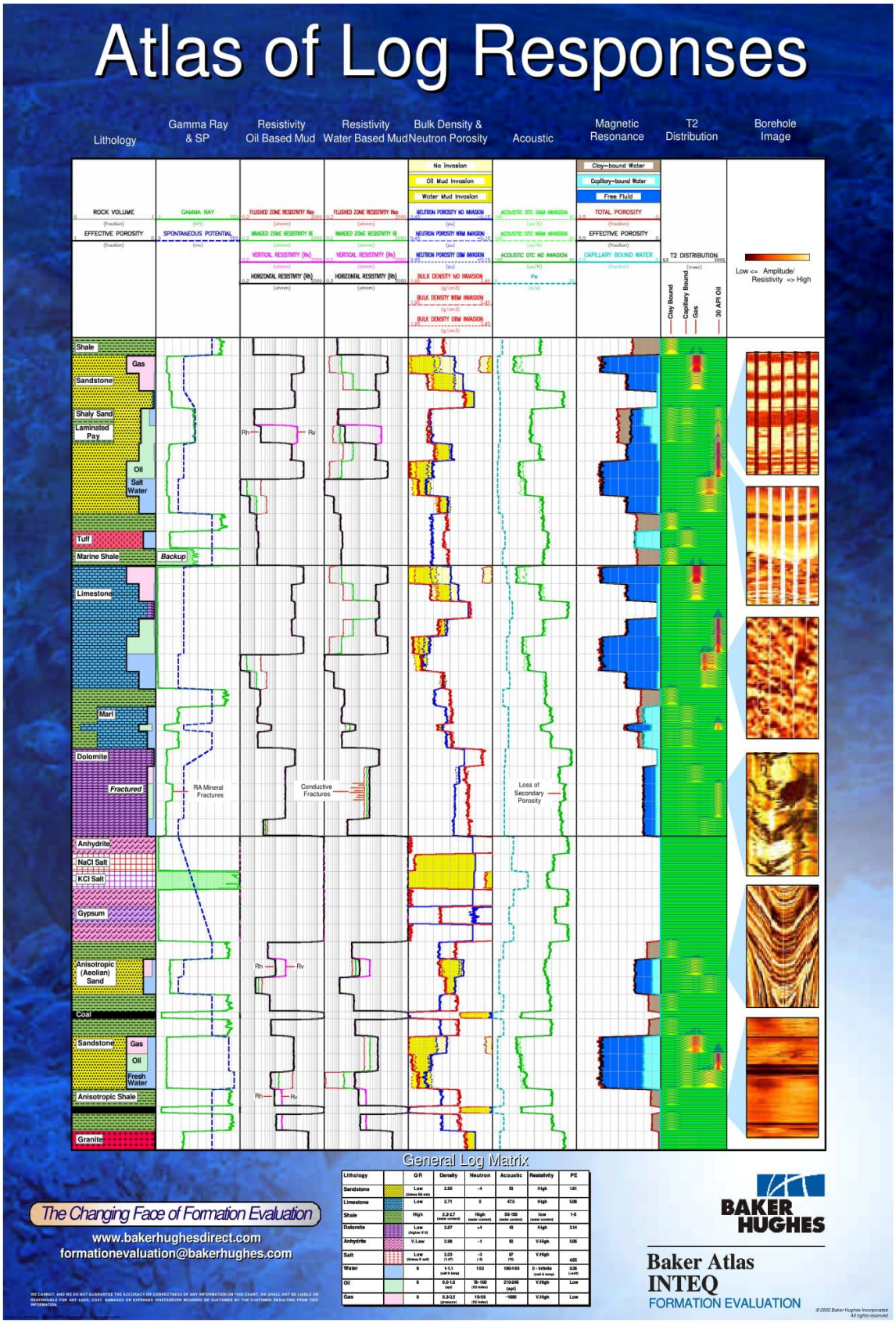


Рисунок 0.1 - Приклад каротажного журналу, на базі атласу гірських порід

РОЗДІЛ 1ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основі визначення

Пласт – геологічне тіло, що має плоску форму, при якій його товщина в багато разів менша за розмір площі його поширення, характеризується однорідними ознаками і обмежене паралельними поверхнями: верхньою — покрівлею та нижньою — підошвою. Товщина пласта визначається за найкоротшою відстанню між покрівлею та підошвою. Пласт може складатися з деяких будь-чим пов'язаних прошарків різних порід (наприклад, пласт вугілля з прошарками аргілітів). Основні параметри пласта та його розташування у просторі характеризується елементами залягання: товщиною (потужністю), кутом падіння, азимутом простягання[1].

Каротаж – геофізичне дослідження свердловин електричними, магнітними, радіоактивними, акустичними та іншими методами з метою вивчення геологічної будови місцевості та виявлення корисних копалин. На практиці застосовують широкий ряд методів проведення каротажу. Їх численність обумовлена різноманіттям методів наземної геофізики, для кожного з яких розроблено аналогічний «підземний» варіант. Більш того, існують і спеціальні види каротажу, що не мають аналогів в наземній геофізиці. Тому методи каротажу розрізняють за своєю природою досліджуваних ними фізичний полів: електричні, ядерні та інші[1].

Літологічна інтерпретація результатів, здебільшого, виконуються складними комп’ютерними програмами, але базовий короткий висновок можна зробити шляхом візуального огляду відповідних журналів.

Найкращі літологічні данні ті, які залежать від властивостей твердих порід, та в меншій степені від властивостей рідин[17]. Найбільш корисні із зазвичай доступних каротажних даних будуть описані далі.

1.2 Гамма каротаж (GR)

Поширені радіоактивні елементи – калій, торій, уран – наявні, зазвичай, в малій кількості у пластових рідинах, тому вони і незначні, тоді як вони є важливою компонентною твердих порід, особливо глинистих мінералів як показано на таблиці 1.1.

Таблиця 1.1 - Таблиця основних відповідей на гамма-випромінювання

Літологія	**Значення гамма-випромінювання, в одиницях API**(American Petroleum Institute – Американський Інститут Нафти)
Пісковик (кварц)	15–30 (рідко до 200)
Вапняк	10–40
Сланці	60–150
Сланці багаті на органіку	100–250
Ангідрит, галіт	8–15
Сильвін	350–500
Доломіт	15–40 (рідко до 200)
Вугілля	15–150 (можливе будь-яке значення)

Форма відповіді гамма-променів (або SP) через піщане тіло часто розглядається як профіль візерунка зерна. Розпізнаються три основні фігури візерунка відповіді: лійка (розширення вгору, зауважте, що це найменший інтервал, тому сланці є найменш ущільненими), циліндр (блочний, зауважте, що журнал SP (Спонтанний потенціал) не відрізняється тим, що засоленість свердловини така ж, як і засоленість пласта) і дзвіночок (звуження вгору,

зауважте, що крім сланців присутнє вугілля) (Рис. 1.1). Ці три форми можна підрозділити на гладкі (відносно однорідні) або зубчасті (з переплетеними тонкими сланцями).

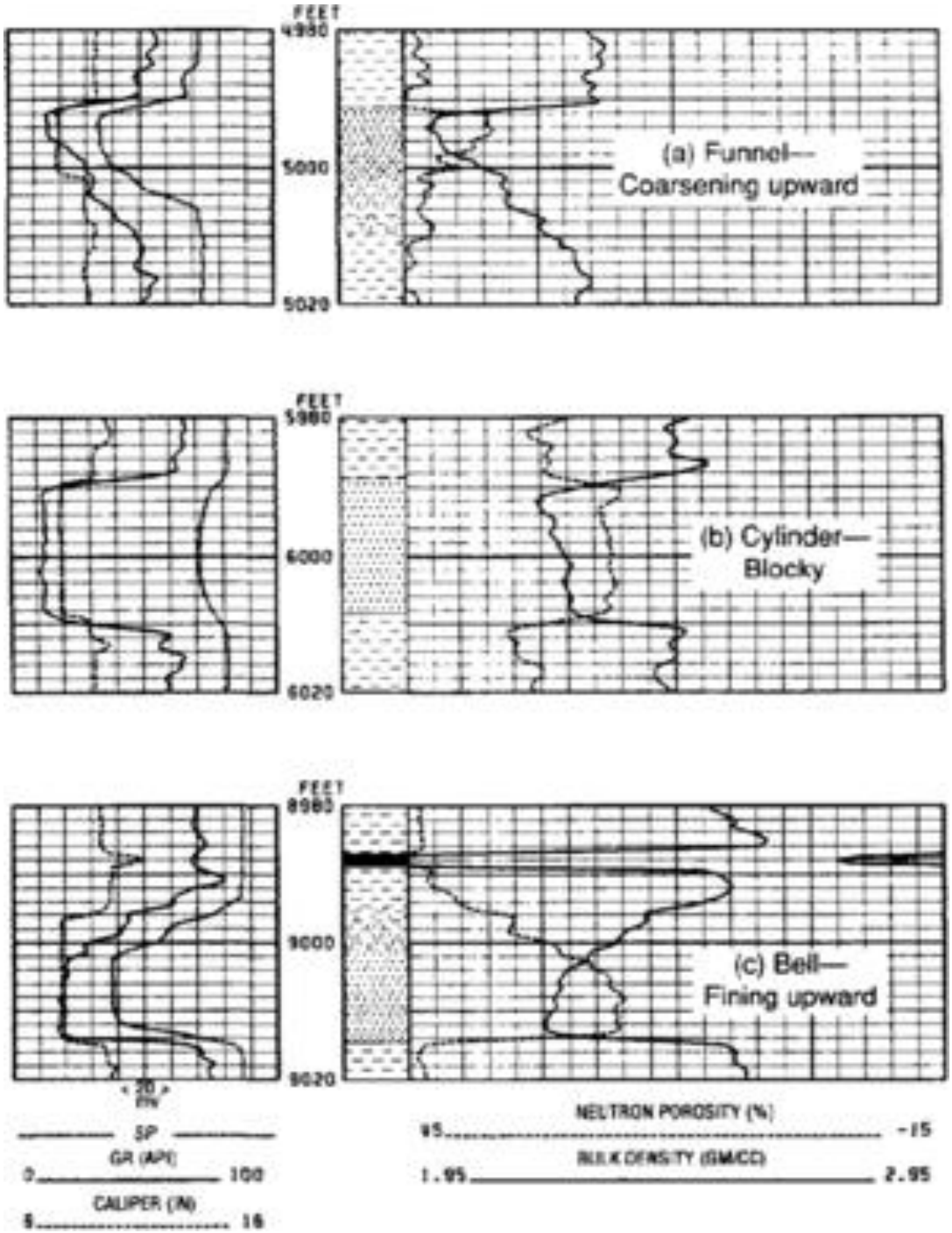


Рисунок 1.1 - Каротаж. Характерні форми для різних типів піщаних тіл, встановлених у сланці, (а) Форма лійки, розширення вгору. (б) Циліндрова форма, блочна. (с) Форма дзвіночка, звуження вгору

Форма відповіді, зазвичай, відображають зміну відкладеної енергії, від високої (чистий, більш грубий пісок) до низької (сланцюватий, більш

дрібний пісок). Інтерпретаційний стрибок, як правило, робиться від енергії депонування до процесу осадження, а отже, і до середовища депонування. Часто цей стрибок робиться без серйозного врахування проміжних кроків. Це може бути небезпечно. Кожен з етапів є дуже неоднозначним і повинен бути доповнений іншими показниками, такими як - щільність часток, пов'язані з ними типи гірських порід та загальні умови осідання.

Зазвичай форма:

– Лійки - означає енергію, що збільшується вгрору, яку можна знайти в вусних смугах - як правило, посеред русла в дельті річки, бахромах дельт, глибоководних океанічних западинах та інших середовищах;

– Циліндра - відображае відносно постійний рівень енергії, яка знаходиться в еолових відкладинах, каналах низької синусоїдальності та розподільних каналах;

– Дзвіночка - представляє собою послідовно зменшувані потоки, які можуть включати алювіальні точкові смуги, дельта-розщирення та глибоководних океанічні западини.

Розмір зерна не впливає на результат гамма-тесту. Форма відповіді відображає вміст сланцю, глини та слюди в піску. Оскільки більшість пісків відображає гідродинамічну рівновагу, вміст глини зазвичай корелює (обернено) з розміром зерна.

1.3 Спонтанний потенціал (SP)

Являється природною різницею потенціалів Землі, вимірюється з допомогою електроду відносно фіксованого опорного електроду. Часто проводять виміри вздовж свердловини для оцінки пластів в нафтових і газових промисловостях.

– Сланці. Інтерпретація спонтанного потенціалу залежить від початкового розпізнавання сланцю, де досить постійні показання SP формують пряму «базову лінію сланця» на каротажі (рис. 2 - а). Його фактичне значення SP не є суттєвим;

– Пісковики. Перепади потенціалів навколо піщано-сланцевого контакту відхиляють SP від базової лінії сланця. Відхилення від'ємне для нормального контрасту солоності. Незначні зміни відбуваються в вздовж пласту с пісковиком, тому чистий пісок показує прямолінійну «піщану лінію» (рис. 2 - б);

– Тверді породи. SP-каротаж мало корисний за відсутності меж між сланцевими пластами та твердими подушками. У відносно щільних гірських породах (карбонати, евапорити тощо) SP блукає безцільно, без різких корисних відповідей.

Форми: лійка; циліндр; дзвіночок, нагадують раніше описані для журналів гамма-променів. Вони пояснюються якісними показниками сланцюватості, наведеними SP, і тому можуть бути інтерпретовані аналогічно гамма-променю.

Контрастна солоність є критичною для журналів SP. Можливі три сценарії. Свіжа свердловинна рідина в сольовому утворенні. Дає "нормальний" SP. Солоність свердловини така ж, як і пластова. Безхарактерний SP, дуже низька амплітуда, може бути прямою лінією, явного відношення до покладів немає (мал. 1б). Сольовий свердловина у

свіжішому пласті. Дає зворотний SP, де пісковики демонструють позитивні відхилення від базової лінії сланців.

1.4 Діаметр свердловини (CALI)

Для літологічних цілей критичними даними є показання розміру отвору залишеного свердловиною, корелює в розміром бітів. Існує три сценарії (таб 1.2).

Таблица 1.1 - сценарій бітів

Тверде тіло	Отвір рівний	Розмір отвору = розмір біту
М'яке або крихке тіло	Вимивається з отвору	Розмір отвору > розмір біту
Проникне тіло	В’язкий бруд	Розмір отвору

Сучасні сенсори дозволяють мінімізувати вплив рідин і видавати більш чіткі значення діаметру отвору як показано на рисунку 1.2:

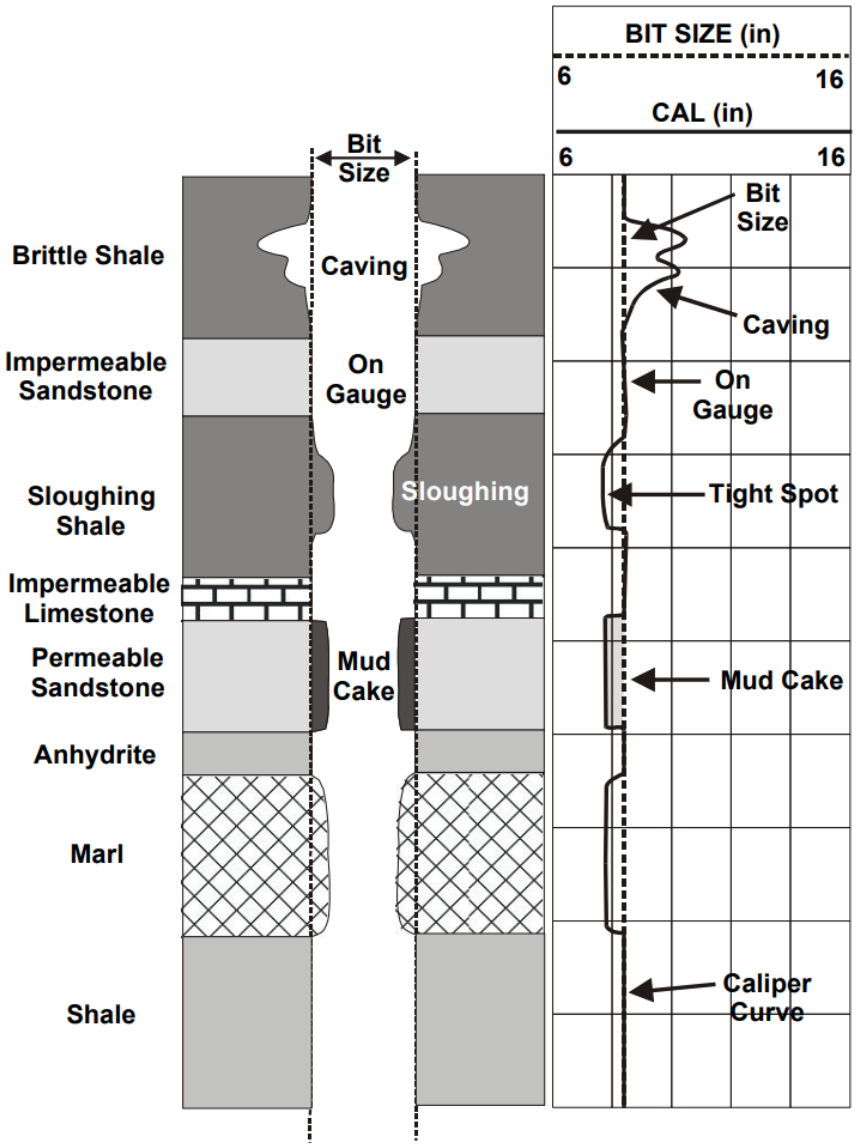


Рисунок 1.2 приклад типових сначень розміру отвору для різних літологічних елементів

– Пісковик. Чистий - без домішків, як правило, проникний , тому рідинний бруд буде наслідком зчитування діаметру на 0.5 дюйма менше за розмір біта. Межі отвору часто точно розмежовані (рис. 2);

– Пісок. Сипкий неконсолідований пісок може вимиватися, викликаючи великі показання;

– Сланець. Часто всипається в свердловину, особливо в мінімальному напрямку основної напруги. Це призводить до еліптичних графіків;

– Вугілля. Вугілля середнього та високого рангу часто крихке і добре з'єднане. Такий вугільний шов, під час роботи свердловини, залишае чіткий товстий слід (рис. 2 - с), але не всі вугілля ведуть себе таким чином;

– Карбонати. Часто не демонструють нарощування грязьового кексу, незважаючи на гарну проникність, оскільки окремі вуглецеві або плісняві пори занадто великі, щоб утримувати бруд;

– Тверді породи. Щільно зацементовані пласти, такі як залізні камені, алевроліти і карбонатні конкременти в пісковиках, є твердими, інертними породами.

1.5 Щільність пласта (DEN)

Пластова щільність - це сума щільності породної системи(з усіх боків пласта) та щільності рідини в породі[18]. Таким чином, значення щільності можна використовувати безпосередньо для ідентифікації літології лише тоді, коли пористість незначна. У пористих породах щільність повинна інтерпретуватися в поєднанні з нейтронними або іншими каротажами пористості як показано на рисунку 1.3:

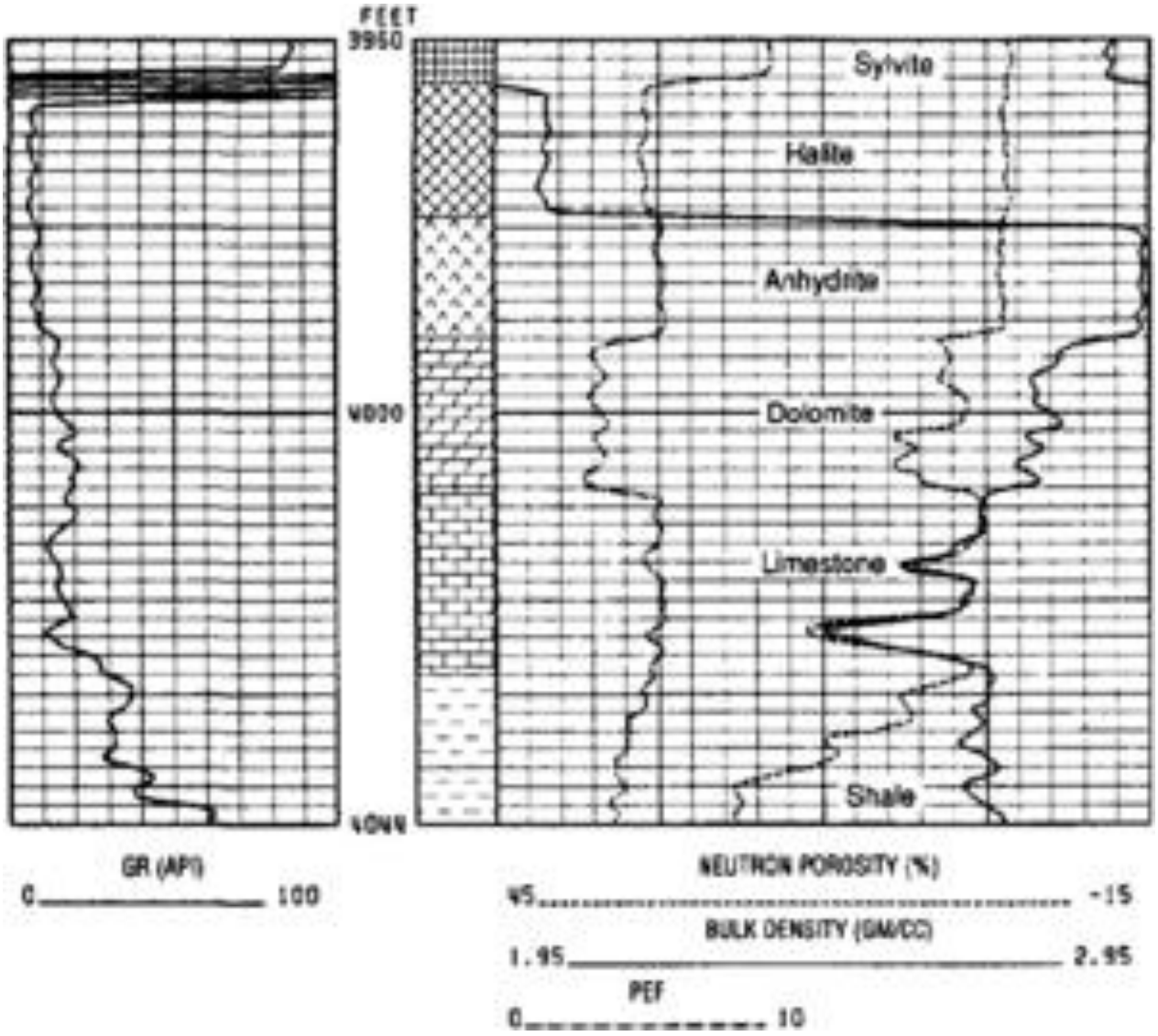


Рисунок 1.3 - Каротаж. Характерні відповіді для карбонатної та евапоритної послідовності. Стан отворів - хороший

– Евапорти. Окремі евапоритні мінерали (такі як ангідрит, галіт, сильвіт і карналіт) мають чітко визначені щільності і створюють прямолінійний каротаж щільності з невеликими варіаціями (рис. 1.3);

– Вугілля. Є змінним, але завжди значно легшим, ніж 2 г/см3. Тонкі подушки дають яскраво виражений сплеск щільності, але можуть спричинити некоректний замір щільності (рис. 1.1 - с);

– Залізний камінь. Концентрації мінералів заліза, таких як пірит і сидерит, дають високу щільність, часто в тонких шарах, контрастуючи з навколишніми породами;

– Сланці. Щільність сланців змінюється в межах від 2,2 до 2,65 г/см3 і більше , збільшуючись при збільшені комплексності, глибини поховання та віку пласта (рис. 1.1). Сланці з надлишковим тиском, в яких частина перевантаженого навантаження покладається на породову рідину, є слабо комплектними і мають низьку щільність відносно звичайних сланців під тиском на однаковій глибині.

1.6 Фотоелектричне поглинання (PE)

Фотоелектричне поглинання (Pe), досить новітній вимір, який фіксується завдяки сучасним сенсорам щільності пласта, пов'язане з атомним числом Z, піднятим до потужності 3,6 (Z3,6). Отже, дуже легкі компоненти (як породова рідина) мають незначний вплив, що робить цей каротаж дуже корисним для літології. На жаль, важкі елементи мають величезний ефект. Таким чином, кілька відсотків заліза маскує основні літологічні відмінності, а барит робить каротаж непридатним для оцінки.

– Пісковик. Кварц повинен читати від 1,7 до 1,8 бн/е, але більшість інших корисних копалин можуть суттєво підвищити цінність. Оскільки вони зазвичай присутні, каротажний запис має обмежене значення;

– Вапняк. Чистий вапняк відповідає приблизно 5,0 бн/е (рис. 1.3);

– Доломіт. Повинен читати близько 3,0 бн/е, що дозволяє легко відрізнити вапняк від доломіту (рис. 1.3), навіть якщо газ присутній у пласті. Зауважимо, що залізо у доломіті збільшує показання, яке нагадують вапняк;

– Сланці. Середнє значення становить 3–3,5 бн/е , але значення до 7 або 8 бн/е можна отримати залежно від вмісту заліза та попутних мінералів.

1.7 Нейронна пористість (NEUT)

Компенсована пористість нейтронів - це насамперед комбінований вміст водню в рідинні породи та гірських породах. Тому літологію можна інтерпретувати безпосередньо з нейтронних значень лише тоді, коли пористість незначна. У пористих породах нейтронний каротаж повинен інтерпретуватися в поєднанні з іншими каротажами, такими як щільність пласта.

Гіпс і ангідрит, типове значення пористості нейтронів у ангідриті ($CaSO_4$) близьке до нуля, але в гіпсі ($CaSO_4 \cdot 2H_2O$) значно вище - до 60%.

Калійні евапорити, сильвіт безводний - майже нульова пористість нейтронів.

Карналіт ($KMgCl_3 \cdot 6H_2O$) - дає значення нейтронів від 30% до 60%.

Деяка вода в сланцях хімічно пов'язана з глинистими мінералами, а частина - з мікропорами. Обидва типи підвищують показання, але не відображають точної пористості (рис. 1.2). Отже, сланці мають високу видиму пористість нейтронів, але значення різняться між утвореннями. Часто, 40% - це хороша межа сланцевого утворення, але значення сланців можуть становити до 30%.

Каротаж нейронної пористості та щільності тісно взаємопов’язані з результатами літології та пористості в цілому, отже аналізуючи ці два каротажі разом, можна вже відрізняти літологію і пористість[11]. Нейтронний та щільнісний каротажі разом з виміром розміру отвору (CALI), зазвичай, разом приймають роль в визначеннях літології. Це найпотужніший із загальнодоступних наборів записів для визначення літології загального призначення.

Набагато швидший спосіб візуалізації типу гірських порід полягає безпосередньо в поданні накладення нейтронного запису та запису густини на один каротаж. Для цього необхідно використовувати сумісну шкалу, щоб

компоненти пористості точно накладалися. Тоді будь-яке зміщення між двома відповідями пояснюється літологічним визначенням або наявністю газу.

Обидва інструменти, як правило, калібруються у вапнякових одиницях, тому сумісна шкала визначається для прісноводних вапнякових систем із теоретичними межами як показано на таблиці 1.2:

Таблиця 1.2 - Сумісна шкала. Для вапнякових систем

	Повна пористість (H_2O)	Без пористості ($CaCO_3$)
Нейтронний (v/v)	100.0	0.0
Щільності (г/см3)	1.0	2.71

На практиці рідко потрібно розглядати пористість понад 50%, тоді як гірські породи з щільністю понад 2,71 г/см3. Таким чином, при незначному округленні звичайна сумісна шкала як показано на таблиці 1.3:

Таблиця 1.3 - Сумісна шкала. Гірські породи з щільністю понад 2,71 г/см3

Нейтронний (v/v)	45.0	30.0	15.0	0.0	15.0
Щільності (г/см3)	1.95	2.20	2.45	2.70	2.95

У районах з високою пористістю, в яких немає доломіту, шкала часто ковзає до наступного діапазону як показано на рисунку 1.4:

Таблиця 1.4 - Сумісна шкала. Райони з високою пористістю, де немає доломіту

Нейтронний (v/v)	60.0	45.0	30.0	15.0	0.0
Щільності (г/см3)	1.70	1.95	2.20	2.45	2.70

В цих діапазонах зберігається будь-яке зміщення нейтронів та густини незалежно від пористості. Зсуви зумовлені різницею порід у характеристиках густини та нейтронної щільності. Ідеальні показники для трьох основних заповнених рідиною пористих порід є такими:

– Пісковики. Зміщення щільності на 0,05 г/см3 вліво. Нейтронна пористість зміщена приблизно на 3 v/v вправо;

– Вапняк. Щільнісний і нейтронний каротаж накладаються точно;

– Доломіт. Зміщення щільності на 0,175 г/см3 вправо. Нейтронна пористість зміщена на 4–8 v/v вліво;

– Пісковик (заповнений водою або маслом). Чистий кварцовий пісковик дає типовий двовимірний – нейтронний та щільнісний, каротаж зі зміщенням щільності вліво від нейтрона (рис. 1.2). Домішки деякої глини (утворює сланцевий пісковик) підвищує нейтронність, зменшуючи відстань між каротажними записами, або навіть віддзеркалює їх відносно середини сітки запису. Важчі компоненти, такі як слюда, збільшують щільність, зменшуючи відстань перетину записів або навіть перетинає їх, щоб створити поділ. Перевірте спектральний гамма-промінь, щоб виділити наступне:

— Слюда - тільки калієва радіація;

— Циркон (з іншими важкими мінералами) - торієва або уранова радіація;

— Сидерит, пірит тощо - відсутність підвищеної радіації.

– Пісковик (заповнений газом). У порівнянні з пісковиком, наповненим нафтою або водою, нейтронний журнал для заповненого газом пісковика зчитує аж 10–15 одиниць пористості (занадто низький рівень), а значення журнал щільності може бути зниженим приблизно до 0,05 г / см3. Разом ці ефекти спричиняють перехід каротажу з двох до приблизно п’яти масштабних поділів;

– Пісковик (заповнений повітрям). Невуглеводневий газ у пісковику може давати показання нейтронів, близьких до нуля, залежно від залишкової води та вологості в просторі порди;

– Вапняк. Чистий вапняк не має поділу нейтронної щільності (рис. 1.3). Коли нейтронний каротаж відхиляться до більш високих значень, це значить про наявність глини. У заповненому газом вапняку очікується перехрещення, як описано для пісковика, і використовуйте значення Ре 5 для підтвердження вапняку;

– Доломіт. Характерне розділення від чотирьох до шести масштабів роздільної щільністі, праворуч від нейтрона відносно послідовно у чистому доломіті (рис. 1.3). Газ зменшує або усуває поділ, значення Ре дорівнює - 3, це підтвердження доломіту;

– Сланець. Поділ каротажу з нейтроном зліва від щільності, іноді зміщено сильно (рис. 1.2). Часом поділ становить лише три або чотири шкали поділу, які можуть нагадувати доломіт;

– Вугілля. Каротаж нейтрону та густини для вугілля показують дуже високу видиму пористість (рис. 1.2 - в). Вугілля надає помітні прогини, які не нагадують нічого, крім аномальних випадків. Діатоміт має щільність близько 1,4 г/см3 і вимірювання нейтронів приблизно в 60 одиниць пористості, тому має принаймні сім масштабних поділок.

1.8 Питомий опір (RT)

Найбільш прямий спосіб знаходження опору води - отримання зразка пластової води та вимірювання її питомого опору. Однак на практиці це рідко можливо, оскільки зразки пластової води, якщо вони є, незмінно забруднені високим спектром домішків. Перший крок в інтерпретації каротажу SP - це встановлення піщаних та сланцевих пластів. Це довільні межі як показано на рисунку 1.4 Пісочні лінії зазвичай представляють сильний прогин вліво, сланець прогин праворуч як показано на рисунку 1.5 Відхилення ліворуч від так званої сланцевої лінії називають нормальними

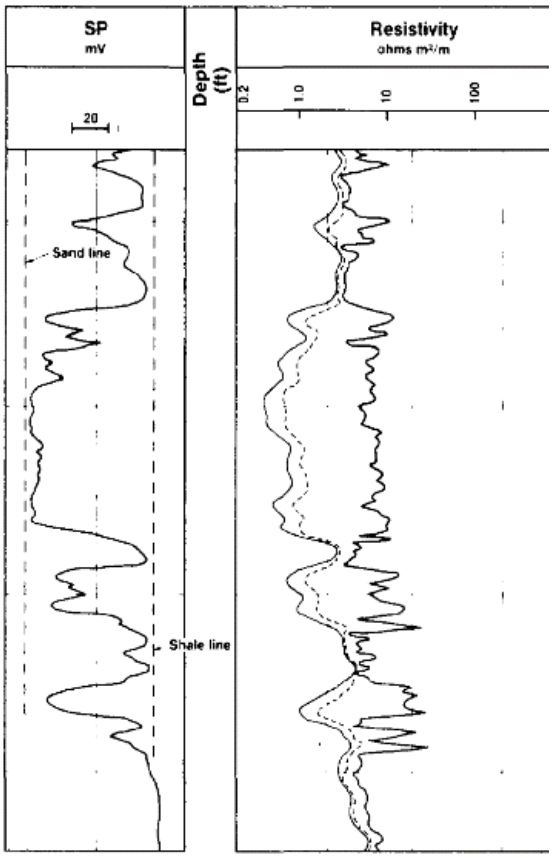


Рисунок 1.4 – Приклад нормального відхилення SP.

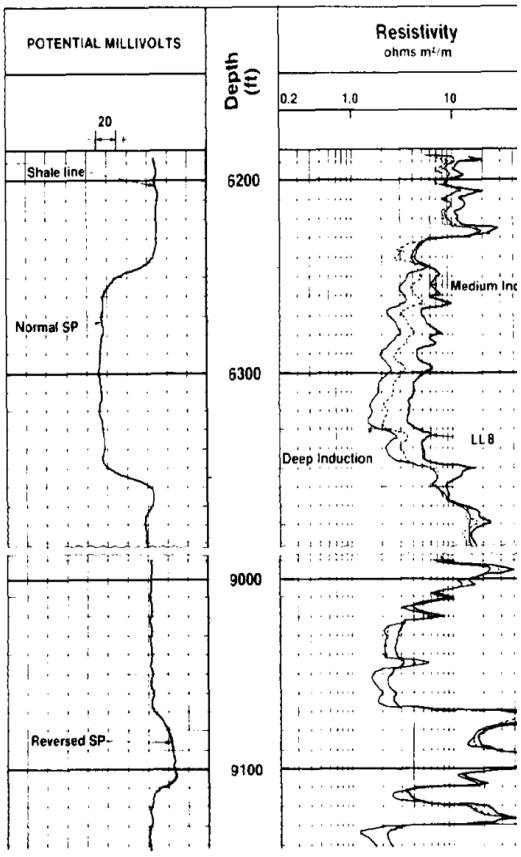


Рисунок 1.5 – Приклад зворотного відхилення SP

або негативними - відповідають пористим і проникним зонам, що містять більш сольову інтерстиціальну воду, ніж бурова грязь[3].

Якщо грязь є більш сольовою, ніж пластова вода, електричні токи, які фіксує SP, течуть у зворотному напрямку і відповідний прогин буде праворуч від сланцевої лінії[2]. Такий прогин вважається зворотним або позитивним (рис. 1.5). Якщо між грязю і пластовою водою немає контрасту солоності, тобто не утворюються електричні течії - відхилення не спостерігається (SP = 0)[2].

Величину відхилення від основної лінії сланців до максимального прогину, що розвинувся в густому чистому водоносному піску, називають статичним SP, або SSP.

1.9 Акустика (DT)

Акустичний каротаж базується на реєстрації швидкості, амплітуди та інших параметрів пружних хвиль ультразвукового і звукового діапазону[7]. Цей метод класифікації літології є найновітнішим і його показники дуже сильно залежить від техніки (сенсору), його типу, виробника, налаштування частотного діапазону і так далі (в даній роботі, разом з вхідними даними не було надано).

Акустичний каротаж виміряється за час блукання звукової хвилі, показання якої записують в мікросекундах на метр або фут[8]. Сенсор випромінює звуковий імпульс, десь один чи два рази в секунду, від трансмітера. Час за який доходить хвиля від емітера до трансмітера – і є шуканим значенням. Сучасне покоління акустичних каротажів може використовувати значення як першої досягнутої звукової хвилі, так і синхронізоване зчитування декількох хвиль – поперечна кореляція, для отримання одного значення (такий підхід дає найбільш точний результат)[9]. В поперечному кореляційному підході застосовуються різного типу хвилі для

різних речовин, які налаштовані на різні частотні діапазони: зсуву породи; кам’яної шару; грязьового кексу; тиску.

Деякі акустичні журнали показують масштаб швидкості, часто нелінійний. Інші репрезентації каротажів можуть використовувати зображення звукових даних як його еквівалентну пористість, перекладені з особливим літологічним припущенням[10]. Налаштування (ваги) зазвичай називають пісковиковими або вапняковими масштабами, щоб відобразити припущення, яке було зроблено для їх створення. Наприклад поклади доломіту можуть буди пере масштабовані в наш карогтаж DT за такими формулами:

$$PHIS = \frac{DELT - KS6}{KS7 - KS6} \ \#(1.1)$$

$$DT = PHIS * KS7 + (1 - PHIS) * KS6 \ \#(1.2)$$

де налаштування:

KS6 = 182 для пісковиків;

KS6 = 155 для сланців;

KS6 = 144 для доломіту;

KS7 = 6.

1.10 Висновки до розділу 1

Геофізичні дослідження свердловин є невід'ємним етапом геологічних, бурових та експлуатаційних робіт, що проводяться при пошуках, розвідці і розробці нафтогазових родовищ. Для отримання різнобічної інформації про геологічну будову надр. Комплексна інтерпретація даних досліджень

повинна охоплювати розрізи всіх свердловин і кожну з них від гирла до вибою.

Інтерпретація матеріалів геоінформаційних досліджень - це творчий процес, глибина якого залежить від обсягу фактичних відомостей про досліджуваному геологічному об'єкті. Детальне вивчення результатів обробки геофізичних даних дозволяє з'ясувати літофаціальную

мінливість відкладень, умови накопичення опадів і формування підняттів. Тому в роботі розглядаються алгоритми розпізнавання об'єктів за допомогою каротажних матеріалів даних

характеристики об'єднання завдань литологической ідентифікації в задачу літологічної інтерпретації.

РОЗДІЛ 2МАТЕМАТИЧНЕ ОБҐРУНТУВАННЯ ВИКОРИСТОВУВАНИХ МЕТОДІВ КЛАСТЕРИЗАЦІЇ

2.1 Вступ до розділу 2

Завдання кластеризації на відміну від інших відомих задач машинного навчання, таких, наприклад, як завдання регресійного аналізу та завдання класифікації, передбачає так зване «навчання без учителя»[19].

Кластеризація застосовується для того, щоб досягти зменшення обсягу даних, скоротити час та ресурси обробки данних, що надходять у поточному часі, за рахунок того, що всі об’єкти так званих вже навчених кластерів, можна розглядати як один об'єкт. Кластеризація може забезпечити кращу інтерпретацію даних у поняття які присутні в темі, що розглядається, та спростити їх подальше опрацювання. Результатом кластеризації є вектор розмірність якого дорівнює кількості вхідних елементів, в свою чергу значення елементів цього вектора – мітка (ідентифікатор, цілочисельне значення область визначення якого від 0, а в деяких методах від -1 – якщо точка не потрапила до жодного з кластерів, до максимальної кількості кластерів, що дорівнює кількості точок)

Тому задача автоматичної та якісно підібраної кластеризації даних за прийнятний час, без попередньої обробки та дослідження інформації, є задачею, дуже важливою та доречною в гірничодобувних, і не тільки, галузях.

2.2 Алгоритм K-Means

Напевне один із найпопулярніших алгоритмів кластеризації. Його алгоритм можна дуже легко описати:

1. Спочатку вибираємо кількість кластерів – k;
2. В просторі наших даних задаємо, випадковим чином, k точок (центроїдів);
3. Для кожної точки, що взялися з вхідного набору даних, рахуємо відстані до центроїдів і визначаємо до якого вона знаходиться найближче;
4. Ідентифікувавши кожну точку – віднесли її до якогось із кластерів, перераховуємо координати центрів кластера, переміщаючи їх в центри вібірок;
5. Повторюємо пункт 3 і 4 фіксовану кількість кроків, чи до тих пір поки центроїди не зійдуться, тобто їх зміщення буде не значне (не більше заданого E - епсилон).

Зазвичай метод пробігає в евклідовому просторі, але алгоритм буде сходитись – завершуватись за скінченну кількість кроків, в будь-якій іншій метричній системі, тому для різних задач в залежності від даних можна експериментувати не тільки з критерієм збіжності та кількістю кроків.

Метод дуже чутливий до початкового розташування центроїдів в просторі. Для біль точних результатів метод запускають декілька разів і беруть усереднений результат кластеризації.

На відміну від задач класифікації або регресії – важче вибрати критерій за яким можна було б інтерпретувати задачу кластеризації як задачу оптимізації , простими словами – вибір оптимального числа кластерів для методу.

$$J(C) = \sum_{\{k=1\}}^{K} \sum_{\{i \sim \in \sim C_k\}} \left|\left|x_i - \mu_k\right|\right|^2 \rightarrow \min_C, \quad \#(2.1)$$

де C – множина кластерів потужності K;

μ_k – центроїд кластера C_k.

Нам потрібно щоб точки знаходили кучно до центра кластерів, але мінімум такої функції буде досягатися тоді, коли кластерів стільки ж як і точок. Для знаходження оптимально числа кластерів, часто користуються таким методом - вибирають те число кластерів, при якому функція $J(C)$ спадає не так швидко:

$$D(k) = \frac{|J(C_k) - J(C_{k+1})|}{|J(C_{k-1}) - J(C_k)|} \rightarrow \min_k, \ \#(2.2)$$

Складність методу для розмірності m, числа кластерів k та числа точок n – $O(n^{dk+1})$. Для дуже великого обсягу даних використовують модифікацію методу - MiniBatch K-means, яка використовує не весь набір для так званого навчання, а лиш деякі його частини.

2.3 Алгоритм Affinity Propagation

На відміну від алгоритму K-Means, даний метод не потребує заздалегідь визначеного числа кластерів. Основна ідея полягає в тому, щоб точки об’єднувались в групи кластерів на основі подібності самих значень.

Для віднесення об’єкту до якогось класу вводиться метрика схожості $s(x_i, x_j)$, наприклад від’ємний квадрат відстані $s(x_i, x_j) = -\left|\left|x_i - x_j\right|\right|^2$, яка значитиме, що спостереження x_i більше схоже на спостереження x_j, ніж на x_k:

$$s(x_i, x_j) > s(x_i, x_k)$$

Для цього зводяться дві ініціалізовані нулями матриці. Перша - $r_{i,k}$, описує наскільки добре k-те спостереження підходить для співвіднесення і-того спостереження на відміну віл всіх інших потенційних прикладів. Друга - $a_{i,k}$, описує наскільки правильно було вибрано k-те спостереження, для і-того в якості прикладу для наслідування.

Матриці оновлюються по черзі за такими правилами:

$$r_{i,k} \leftarrow s(x_i, x_k) - \max_{k' \neq k}\{a_{i,k'} + s(x_i, x'_k)\}, \forall i, k \ \#(2.3)$$

$$a_{i,k} \leftarrow \min\left(0, r_{k,k} + \sum_{i' \notin \{i,k\}} \max\left(0, r_{i',k}\right)\right), \forall i, k; i \neq k \ \#(2.4)$$

$$a_{k,k} = \sum_{i' \neq k} \max\left(0, r_{i',k}\right), \forall k \ \#(2.5)$$

$$c_i \leftarrow arg\max_k\left(a(i,k) + r(i,k)\right), \forall i \ \#(2.6)$$

Як і в багатьох інших алгоритмах виконання Affinity propagation можна перервати, якщо матриці R і K перестають оновлюватися. Для цього, в програмній реалізації цього методу використовуються два параметри: максимальна кількість ітерацій та період, за який перевіряється величина зміщення від попередніх підрахунків.
З самого початку до матриці подібності – R, додають трохи шуму, щоб уникнути детермінованого зациклення. Також при оновлені матриць використовується не просте присвоєння, а з експоненціальним згладжуванням.

$$S_t = \alpha x_t + (1 - \alpha)S_{t-1}, \alpha \in (0,1) \ \#(2.7)$$

Це добре впливає на якість результату, але ця оптимізація збільшує необхідну кількість ітерацій T.

Складність методу - $O(n^2T)$, де N – кількість об’єктів набору даних, а T – кількість ітерацій, та займає $O(N^2)$ пам’яті.

2.4 Алгоритм Spectral Clustering

В алгоритмі використовується матриця спостережень. Вона обчислюється за спектром нормалізованої матриці Лапласа. Інформація про структуру кластерів знаходиться у власному векторі матриці. На вхід алгоритма надходять точки, розмічені з допомогою метода KLT(Алгоритм Лукаса - Канаде)[4]. На основі виділених точок будується зважений граф. Для опису графа, де кожна вершина відповідає точці з набору даних, а кожна пара вершин з’єднана ребром. Необхідно побудувати матрицю збіжності. Матриця збіжності вираховується з використанням метрики Гауса:

$$W_{ij} = \exp\left(-\frac{\left||x_i - x_j|\right|^2}{2\sigma}\right) \#(2.8)$$

де x_i, x_j – точки з набору даних між якими вираховується подібність;

σ – параметр з допомогою якого регулюється чутливість функції подібності.

З допомогою матриці W вираховуємо матрицю степенів графа і на цьому ж етапі вираховується сума степенів вершин:

$$vol = \sum_{v \in V} d_v \ \#(2.9)$$

де d_v – степінь вершини графа.

Для підрахунку випадкового блукання графа і отримання матриці часу блукання використовують нормалізовану матрицю Лапласа. Для зваженого графа нормалізована матриця Лапласа має вигляд:

$$L_n = \begin{cases} 1, u = v; \\ -\dfrac{w(u,v)}{\sqrt{d_u d_v}}, u \neq v; \\ 0, \text{в іншому випадку}; \end{cases} \quad \#(2.10)$$

де w(u,v) – вага ребра, яке з'єднує вершини u та v;

$d_u d_v$ – степені вершин.

Для вирахування власних значень і власних векторів нормалізованої матриці Лапласа необхідно провести декомпозицію :

$$L_n = \phi \Lambda \phi^T \ \#(2.11)$$

де Λ – діагональна матриця власних значень $\lambda_1, \lambda_2, \dots, \lambda_{|V|}$;

ϕ – матриця власних векторів.

Функція Гріна зв'язана з матрицею Лапласа, характеризує зв'язок вершин графа і використовується для підрахунку матриці часу обходу. На основі нормалізованої матриці Лапласа і її спектра, можна отримати матрицю Гріна:

$$G_{ij}(u,v) = \sum_{i=2}^{|V|} \frac{1}{\lambda_i} \varphi_i(u) \varphi_i(v) \ \#(2.12)$$

де λ_i і φ_i – власне значення і вектор відповідно, нормалізованої матриці Лапласа L_n.

Припустимо, що H(u,v) – час обходу випадкового шляху між вершинами u та v. Тоді час обходу буде мати вигляд $CT(u,v) = \mathrm{H}(\mathrm{u},\mathrm{v}) + \mathrm{H}(\mathrm{v},\mathrm{u})$.

Матриця часу підраховується с використанням матриці Гріна:

$$CT(u,v) = vol\sum_{i=2}^{|V|}\frac{1}{\lambda_i}\left(\frac{\varphi_i(u)}{\sqrt{d_u}} - \frac{\varphi_i(v)}{\sqrt{d_v}}\right)^2 \quad \#(2.13)$$

Елементи цієї матриці являються значеннями часу обходу між кожною парою вершин. Далі відбувається спектральна декомпозиція матриці часу обходу. Елементи власного вектору $CT\varphi_0$, відповідають мінімальному власному значенню цієї матриці та містять інформацію про кластерну структуру графа. Кожне значення цього вектора характеризує відповідну точку вихідного простору даних. Віднесення точки до кластерів відбуваються за наступною умовою:

$$Point(v)\epsilon\begin{cases} Cluster1, CT\varphi_0(v) \geq 0; \\ Cluster2, \text{в іншому випадку;} \end{cases} \#(2.14)$$

Складність методу - $O(n^{3/2})$.

2.5 Алгоритм Spectral Biclustering

Алгоритм передбачає, що матриця вхідних даних має приховану контрольну структуру. Рядки та стовпці матриці з цією структурою можуть бути розподілені таким чином, що записи будь-якого бікластеру в декартовому добутку кластерів рядків та кластерних стовпців є приблизно постійними. Рядки та стовпці вхідної матриці розбиваються на постійну контрольну блок-матрицю яка забезпечує гарну апроксимацію вхідної

матриці. Вхідна патриця спочатку нормалізується щоб зробити об’єкти контрольної матриці біль очевидними. Є три можливих методи:

1. Незалежна нормалізація рядків та стовбців, як і в алгоритмі Spectral Co-Clustering. Цей метод робить щоб сума рядків дорівнювала одній константі, а сума стовбців різним константам;

2. Бістохастизація: повторна нормалізація рядків і стовпців до конвергенції. Цей спосіб змушує суму рядків та стовпців дорівнювати однаковій константі;

3. Логарифмічна нормалізація: рахується логарифм від вхідної матриці $L = logA$. Потім вираховуються середні значення стовбців - $\overline{L_i}$, та рядків - $\overline{L_j}$, і загальма матриця середніх значень - $\overline{L}$. Фінальна матриця рахується за формулою:

$$K_{ij} = L_{ij} - \overline{L_i} - \overline{L_j} + \overline{L} \#(2.15)$$

Після нормалізації підраховуються декілька перших сингулярних вектори, точно так як і в алгоритмі Spectral Co-Clustering.

Якщо була застосована логарифмічна нормалізація, всі сингулярні вектори визначено. Проте, якщо була застосована незалежна нормалізація або бістохастична перші сингулярні вектори u_1 і v_1 відкидаються. Від тепер за початкові сингулярні вектори відповідають $u_2 \dots u_{p+1}$ і $v_2 \dots v_{p+1}$ (крім логарифмічного випадку).

З огляду на ці особливі вектори, вони ранжуються, згідно з тим які з них можна найкраще можна апроксимувати до кусочно-постійного вектора. Наближення для кожного вектора вираховують з допомогою одновимірного k-mean алгоритму та оцінюються за допомогою евклідової відстані. Вибирають деяку підмножину найкращого лівого та правого сингулярного вектора. Далі дані проектуються на цю найкращу підмножину сингулярних векторів та кластеризуються.

Наприклад, якщо p – вирахуваний сингулярний вектор, q – найкращий з вирахуваних, тобто $q < p$. Нехай U – матриця з сингулярними векторами в колнках і найкращий – q, знаходиться зліва, і така ж матриця V, але справа. Щоб розділити строки, строки матриці A проектують на простір розмірності q: $A * V$. Ці m рядків матриці m на q, як подаємо на вхідні методу K-Means і кластеризуємо ці рядки.

2.6 Алгоритм Agglomerative Clustering

Один із самих простих і зрозумілих алгоритмів кластеризації, наперед заданого числа кластерів в параметрах.

1. Спочатку ідентифікується кожна точка як центр нового кластеру;
2. Відсортовуються по зростанню попарні відстані між центрами кластерів;
3. Пара найближчих кластерів склеюється о дин і перераховується його центр;
4. Пункти 2 і 3 повторюються до тих пір поки всі дані не склеються в один кластер.

Процес пошуку найближчих кластерів може відбуватися з використанням різних методів склеювання точок:

1. Мінімум попарних відстаней між точками двух кластерів –

$$d(C_i, C_j) = min_{x_i \in C_i, x_j \in C_j} \left|\left|x_i - x_j\right|\right| \quad \#(2.16)$$

2. Максимум попарних відстаней між точками двух кластерів –

$$d(C_i, C_j) = \max_{x_i \in C_i, x_j \in C_j} \left|\left|x_i - x\right|\right| \quad \#(2.17)$$

3. Центр попарних відстаней між точками двух кластерів -

$$d(C_i, C_j) = \frac{1}{n_i n_j} \sum_{x_i \in C_i} \sum_{x_j \in C_j} \left|\left| x_i - x_j \right|\right| \quad \#(2.18)$$

4. Відстань між центроїдами двух кластерів -

$$d(C_i, C_j) = \left|\left| \mu_i - \mu_j \right|\right| \quad \#(2.19)$$

Перевага перших трьох методів на відміну від четвертого в тому, що в них не потрібно перераховувати відстані кожен раз після склеювання, що дуже знижує облікову складність алгоритму.

За результатами виконання можна або, побудувавши дерево склеювання, графо-аналітично визначити оптимальну кількість кластерів, або застосувати метод як в алгоритмі K-Means.

2.7 Алгоритм DBSCAN

DBSCAN (Density-based spatial clustering of applications with noise) – шільнісний алгоритм просторової кластеризації з шумом. На вхід подається матриця схожості і два параметри – радіус околу – ϵ, і кількість сусідів - m.

Нехай задана деяка функція відстані $\rho(x, y)$ і константи ϵ та m, тоді:

1. Назвемо простір $E(x)$, для якого $\forall y$: $\rho(x, y) \leq \epsilon$, ϵ-окіл об'єкта x;
2. Корневим об'єктом степені m називається об'єкт, ϵ-окіл якого включає не менш ніж m об'єктів:$|E(x)| \geq m$;
3. Об'єкт p досягається об'єкта q за щільністю, якщо $p \in E(q)$ та q - кореневий об'єкт;
4. Об'єкт p досягає за щільністю об'єкт q, якщо $\exists p_1, p_2 \dots p_n, p_1 = q, p_n = p$, такі що $\forall i \in 1 \dots n-1$: p_{i+1}безпосередньо досягається за щільністю із p_i.

Виберемо який-небудь кореневий об'єкт p із набору даних, відмітимо його і помістимо всіх його сусідів, за щільністю, в список для подальшого розгляду. Тепер для кожного q із списку: відмітимо цю точку і, якщо вона теж коренева, додаємо всіх її сусідів в список для подальшого розгляду. Точки цього кластеру максимально відмічені, тобто неможна віднести ще якусь точку до цього кластеру, щоб потрапляла в щільність цього кластеру та задовольняла умові. Тобто, якщо всі точки від корінного об'єкту точки розмічені, то запустивши нову ітерацію формування кластеру від нового корінного об'єкту, він не поглине попереднього кластера. Приклад алгоритму на базі псевдокоду:

I. Для кожної точки з набору даних:
   a. Якщо вже відмічена як вже «відвідувана» – наступна ітерація;
   b. Відмічаємо точку як «відвідувана»;
   c. Беремо всі точки що лежать в епсилон околі - ϵ, включаючи саму точку;

d. Якщо кількість точок в околі, включаючи саму точку, строго менше за заданий параметр m – відмічаємо точку як «шум». В іншому випадку переходимо до етапу II.

II. Задається новий кластер. Додаємо точку до цього кластеру. Для кожної точки в околі (визначених ще в пункті I):

a. Якщо точка не відмічена як «відвідувана»:

1. Відмічаємо її як «відвідувана»;

2. Беремо точки з околу ϵ, включаючи саму точку;

3. І якщо кількість точок в околі ϵ, включаючи саму точку більше або дорівнює параметру m – об'єднуємо точки з околу цієї точки разом з околом визначними на єтапі I, тобдо ми додали ці точки в список потенціальних точок для створеного кластеру.

b. Якщо точка ще не відмічена як член створеного кластера – включаємо цю точку в цей кластер.

В більшості реалізацій, крайові точки потрапляють до першого кластеру який її досягнув. Як правило це не сильно впливає на якість роботи алгоритму, адже через граничні точки кластеру подальше віднесення інших точок околу цієї граничної точки не відбувається – це неможливо.

Співвідношення $\frac{m}{\epsilon^n}$ – гранична щільність точок в просторі, де n – розмірність простору. Очікується, що при однаковому співвідношенні $\frac{m}{\epsilon^n}$ – результат кластеризації буде дуже схожим. Чим більшого значення набувають гіпер-параметри, тим більше алгоритм допускає меншу щільність точок в кластері. Іноді це корисно, коли між даними є пропуски але бажано щоб об'єкти були співвіднесені до одного кластеру, а іноді, коли між даними немає чітких границь, так звані шуми можуть спричинити «міст» між даними які не повинні були б бути в спільному кластері. В правильному підборі гіпер-параметрів якраз і ховається складність методу.m n

Для підбору параметрів m та ϵ можна скористатися таким евристичним алгоритмом:

1. Спершу вибирати m, зазвичай його беруть в діапазоні - (3, 9), чим більш є неоднорідний очікуваний набір даних і чим більший рівень шуму, тим більшого значення повинен набувати гіпер-патраметр m;

2. Вирахувати середню відстань до m найближчих сусідів для кожної точки;

3. Сортуємо отримані результати за зростанням та будемо графік (вісь абсцис – відстані між всіма точками попарно, вісь ординат – середня відстань між m найближчими сусідами);

4. Графік матиме вигляд схожий до експоненти - e^x, слід вибрати ϵ десь в інтервалі різкого перегину (значення беруть по осі ординат). Чим більше ϵ тим більше кластери і тим менша їх кількість.

DBSCAN самостійно не вираховує центри кластерів, але автоматично ідентифікує шуми.

Складність методу: в ідеальному випадку – $O(n)$; якщо не перераховувати кожен раз простір $E(x)$, то очікувана складність - $O(n \log n)$; найгірший випадок - $O(n^2)$.

2.8 Алгоритм OPTICS

OPTICS (Ordering points to identify the clustering structure) – це алгоритм знаходження кластерів в просторі даних на основі щільності[14]. Алгоритм цього методу дуже схожий на метод DBSCAN, але головна його відмінність в тому, що він бере до у ваги і намагається вирішити один з головних недоліків DBSCAN – погане виявлення кластерів з різними щільностями. Для цього будується «граф доступності», для побудови якого, вхідні дані – точки, лінійно сортуються за зростанням, але так щоб сусідні точки залишалися сісідніми, та для кожної точки вираховується додаткова щільнісна відстань за

якою точка буде розглядатися до належності тому чи іншому кластеру схожої щільності.

Так само як і DBSCAN – має два основних параметри ϵ і m – максимальний радіус в якому розглядаються точки і мінімальна кількість точок для включення точки в існуючий кластер або утворення нового відповідно. Точка p – вважається точкою для включення в той чи інший кластер, якщо хоча б m точок знаходяться в її околі - $N_\epsilon(p)$, радіусом ϵ. Алгоритм також бере до розгляду точки які належать іншим кластерам з більшою щільністю. Кожній точці вираховується глобальна відстань, яка означає відстань до найближчої точки з m – сусідства:

$$globalDist_{\in,m} = \begin{cases} \text{не визначено}, |N_\epsilon(p)| < m \\ th_m\big(N_\epsilon(p)\big), |N_\epsilon(p)| \geq m \end{cases} \quad \#(2.20)$$

де $th_m(N_\epsilon(p))$ = m-ній відстані до $N_\epsilon(p)$, в порядку зростання. Також вираховується відстань досягаємості для точки q до точки p, яка дорівнює або відстані між цими двома точками, або глобальній відстані точки p, це залежить від того – яка відстань більша:

$$reachDist_{\in,m}(q,p) = \begin{cases} \text{не визначено}, |N_\epsilon(p)| < m \\ \max\Big(globalDist_{\in,m}(p), \quad Dist(p,q)\Big), |N_\epsilon(p)| \geq m \end{cases} \quad \#(2.21)$$

Якщо p і q являються найближчими сусідами та $\in' < \in$, робиться висновок, що p і q належать до спільного кластеру.

Складність методу схожа на DBSCAN - $O(n\log(n))$.

2.9 Оцінки методів кластеризації

2.9.1 Індекс Calinski–Harabasz

Індекс Калінській-Харабаш набуває більшого (кращого) значення коли кластери щільні і чітко виділені[6]. Цей індекс ефективності кластеризації в більшій степені нагадує те, як людина оцінює результат кластеризації. Результат вважається хорошим, якщо візуально створені чітко виділені і щільно зосереджені групи міток. Індекс використовує два значеня:

– W_k – дисперсія всередині кластера, матриця відстаней між елементами кластера та центром кластера;

– B_k – дисперсія між групами, матриця відстаней між центром кластера і центрами всіх інших кластерів.

Для кожного елемента з набору даних E розмірності n, де в результаті роботи методу кластеризації маємо k кластерів, індекс Калінській-Харабаш рахується за формулою:

$$index = \frac{tr(B_k)}{tr(W_k)} * \frac{n-k}{k-1} \ \#(2.22)$$

де $tr(B_k)$ – це слід між групами дисперсій матриці;

$r(W_k)$ – це слід між матрицею дисперсії внутрішніх груп кластеризації.

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T \ \#(2.23)$$

$$B_k = \sum_{q=1}^{k} n_q (c_q - c_E)(c_q - c_E)^T \ \#(2.24)$$

2.9.2Індекс Davies-Bouldin

Краще значення – те що ближче до нуля. Індекс Девіс-Боулдіна означає середню схожість між кластерами, де мірою подібності є відстань між кластерами та розміри самих кластерів[5]. Як правило, вище (краще) значення набувається для округлих форм кластерів, ніж для інших видів. Наприклад для методів кластеризації, які основані на щільності розподілу (наприклад DBSCAN), оцінка буде гіршою.

Індекс схожості визначається між кожним з кластерів $C_i,\ i = 1, \dots, k$ та найбільш схожим C_j. В контексті цього індексу, схожість – це значення R_{ij} яке рахується за формулою:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \ \#(2.25)$$

де s_i – середня відстань між кожною точкою кластеру i і центром цього кластеру (також відомого як «діаметр» кластеру); d_{ij} – відстань між центроїдами кластеру i та j.

Індекс Девіс-Боулдіна рахується за формулою:

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max_{i \neq j} R_{ij} \ \#(2.26)$$

2.9.3Коефіцієнт Silhouette

Коефіцієнт силуету - приклад такої оцінки, де вищий показник коефіцієнта силуету означає модель з більш чітко визначеними межами між кластерами. Коефіцієнт рахується за формулою (для кожної точки своє значення):

$$s = \frac{b - a}{\max(a, b)} \#(2.27)$$

де a – середня відстань між точкою та всіма іншими точками цього кластеру;

b – середня відстань між точкою та всіма точками наступного найближчого кластера.

Коефіцієнт набуває значення в інтервалі від -1 – поганий результат кластеризації, до 1 – для дуже щільних кластерів з чіткими межами. Значення 0 означає шо кластери склеюються дуже сильно.

2.10 Постановка задачі багатокритеріальної оптимізації. Оптимальність по Парето.

Нехай X означає множину допустимих рішень деякої задачі, $x \in X$ – допустиме рішення. Припустімо, що кожне рішення $x \in X$ оцінюється по n критеріям $(n \geq 2)$. Мається дуже корисна конструкція для вирішення багатокритеріальної задачі - поняття оптимальності по Парето.

Нехай $H_i(x), x \in X$ – функція значення якої є оцінкою рішення $x \in X$ по критерію $i, i = \overline{1,n}$. Тоді вектор $H(x) = \big(H_1(x), \dots, H_i(x)\big), x \in X$ – набір

оцінок рішення $x \in X$ по всім критеріям. Припустімо, що ступінь приналежності рішення $x \in X$ зростає разом з зростанням компонентів вектора H, тобто чим більше значення набуває $H_i(x)$, тим краще рішення x за критерієм $i, i = \overline{1,n}$.

Рішення $x^* \in X$ називається парето-оптимальним, якщо не існує іншого рішення $x \in X$, для якого:

$$H_i(x) \geq H_i(x^*), i = \overline{1,n}, \qquad \exists i_0 : H_{i_0} > H_{i_0}(x^*). \ \#(2.28)$$

Іншими словами, якщо $x^* \in X$ – парето-оптимальне рішення, то не існує іншого рішення $x \in X$, яке є кращим рішення x^*, хоча б за одним із критеріїв, а за іншими критеріями не гірше[12].

Для того щоб рішення $x \in X$, було оптимальним по парето , необхідно і достатньо щоб воно було рішенням наступної задачі багатокритеріальної оптимізації:

$$\max_{x \in X} \sum_{i=1}^{n} \lambda_i H_i(x), \ \#(2.29)$$

де параметри λ_i, $i = \overline{1,n}$, задовольняють умовам $\sum_{i=1}^{n} \lambda_i = 1$, $\lambda_i \in (0,1)$.

2.11 Метод зменшення розмірності TSNE.

Метод перетворює розташування споріднених точок у просторі в ймовірності. Спорідненість точок в нормальному евклідовому просторі виступає у ролі Гаусівських об’єднаних ймовірностей, а спорідненість у підпросторах виступає у ролі т-розподілу Стьюдента. Це дозволяє TSNE більш точно зберігати структуру даних і має ряд інших перевал перед іншими існуючими методами пониження розмірності:

– розкриття з урахуванням щільності на одному графіку;

– виявлення даних, що лежать у кількох різних колекторах чи кластерах;

– зменшує результуючу щільність скупчених у центри точок, для кращої візуальної інтерпретації.

У той час коли інші відомі методи – Isomap і LLE, найкраще підходять для згортання єдиного безперервного низько розмірного кластера/скупчення, TSNE зосереджує логіку свого алгоритму на локальній структурі даних. Починається метод с перетворення багаторозмірої евклідової відстані між точками в умовну вірогідність, яке інтерпретується як схожість точок:

$$p_{j|i} = \frac{\exp\left(-\frac{\left||x_i - x_j|\right|^2}{2\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{\left||x_i - x_k|\right|^2}{2\sigma_i^2}\right)} \quad \#(2.30)$$

формула показує наскільки, при гаусівському розподілі навколо,

точка X_j знаходиться близько до точки X_i з середньоквадратичним відхиленням σ_i – вираховується таким чином, щоб області з більшою

щільністю мали меншу дисперсію. Для цього використовується оцінка «перплексії»:

$$Perp(P_i) = 2^{H(P_i)} \#(2.31)$$

де $H(P_i)$ – ентропія Шеннона в бітах.

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i} \#(2.32)$$

Перплексія задається в якості параметра методу і означає оцінку ефективної кількості сусідніх точок для точки X_i. Рекомендується брати значення в інтервалі від 5 до 50[13].

$$q_{j|i} = \frac{\exp\left(-\left||y_i - y_j\right|\right|^2\right)}{\sum_{k \neq i} \exp\left(-\left||y_i - y_k\right|\right|^2\right)} \#(2.33)$$

В якості оцінки якості, з яким $p_{j|i}$ відображає в $q_{j|i}$, використовується дивергенція Кульбака-Лейблера. TSNE мінімізує суму таких відстаней для всіх точок відображення з допомогою градієнтного спуску. Функція втрат:

$$Cost = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log\left(\frac{p_{j|i}}{q_{j|i}}\right) \#(2.34)$$

Алгоритмічний пошук рівноваги робиться з урахуванням моментів:

$$Y^t = Y^{t-1} + \eta \frac{\partial Cost}{\partial Y} + \alpha(t)(Y^{t-1} - Y^{t-2}) \#(2.35)$$

де Y – множина точок відображення X;

η – параметр швидкості навчання;

α – коефіцієнт інерції (наскільки ми хочемо враховувати значення попередніх ітерацій).

TSNE використовує T-розподіл в спроектованому просторі – що допомагає не сильно псувати відносно реальне розміщення точок в просторі. На відміну від гаусівського розподілу, яке використовує звичайний SNE – точки будуть швидко набирати нульове значення схожості – більшість яких будуть «відштовхуватися один від одного».

2.12 Висновки до розділу 2

Спектр застосувань кластерного аналізу дуже широкий: його використовують в археології, медицині, хімії, біології, фінансовому аналізі та управлінні, маркетингу, соціології, геології та інших дисциплінах.

Можна зробити висновок про існування двох важливих вимог до даних - однорідність і повнота. Однорідність - щоб всі об’єкти були однією природи, описувалися подібним набором характеристик.

У сучасній науці застосовується кілька алгоритмів обробки вхідних даних. Аналіз шляхом порівняння об'єктів, виходячи з ознак або порівняння ознак, на основі об'єктів.

РОЗДІЛ 3РОЗРОБКА КОМП'ЮТЕРНОЇ ПРОГРАМИ ДЛЯ АНАЛІЗУ ДАНИХ

3.1 Використані програмні засоби

В процесі розробки використовувалась мова програмування Python версії 3.8. Ця мова програмування була обрана через її актуальність – найрозповсюдженіша мова програмування на даний момент часу, швидкодію та кросплатформність – як в якості інстальованої програми для персонального комп’ютера так і в якості простої в реалізації, завдяки таким платформам як, наприклад, Jupiter, веб-версія продукту (існує можливість використання як в локальній мережі, так і реалізація користувач-сервер). Простота та водночас широкий спектр можливостей – завдяки вже реалізованих та неоднократно випробуваних на точність та швидкість роботи основного математичного функціоналу та функціоналу обробки і відображення даних, реалізованих в основних модулях цієї мови, дозволяє швидко і легко навчити нових користувачів і надає їм всі можливості для подальшого опрацювання та удосконалення програмного продукту. Найважливіші модулі: pandas та numpy, що мають широкий вбудований функціонал для швидкої обробки великих об’ємів інформації; sklearn – реалізація методів кластеризації, мертрики для оцінок якості кластеризації та методи пониження розмірності даних для кращої візуальної інтерпретації; seaborn та matplotlib – дуже сильні інструменти для побудови різноманітних графіків.

3.2 Вхідні данні та їх аналіз

Данні представляють собою записи з роботи однієї свердловини (аналізуючих приладів), що є не дуже добре, адже якість заключення буде залежати від якості однієї вибірки даних, тобто якщо заміри були проведені не правильно, або з тих чи інших причин робота сенсорів було скомпрометоване – це не можливо буде виявити. Тому вибірка вважається еталонною (що в реальності може бути не так, через описані вище фактори) як показано на рисунку 3.1:

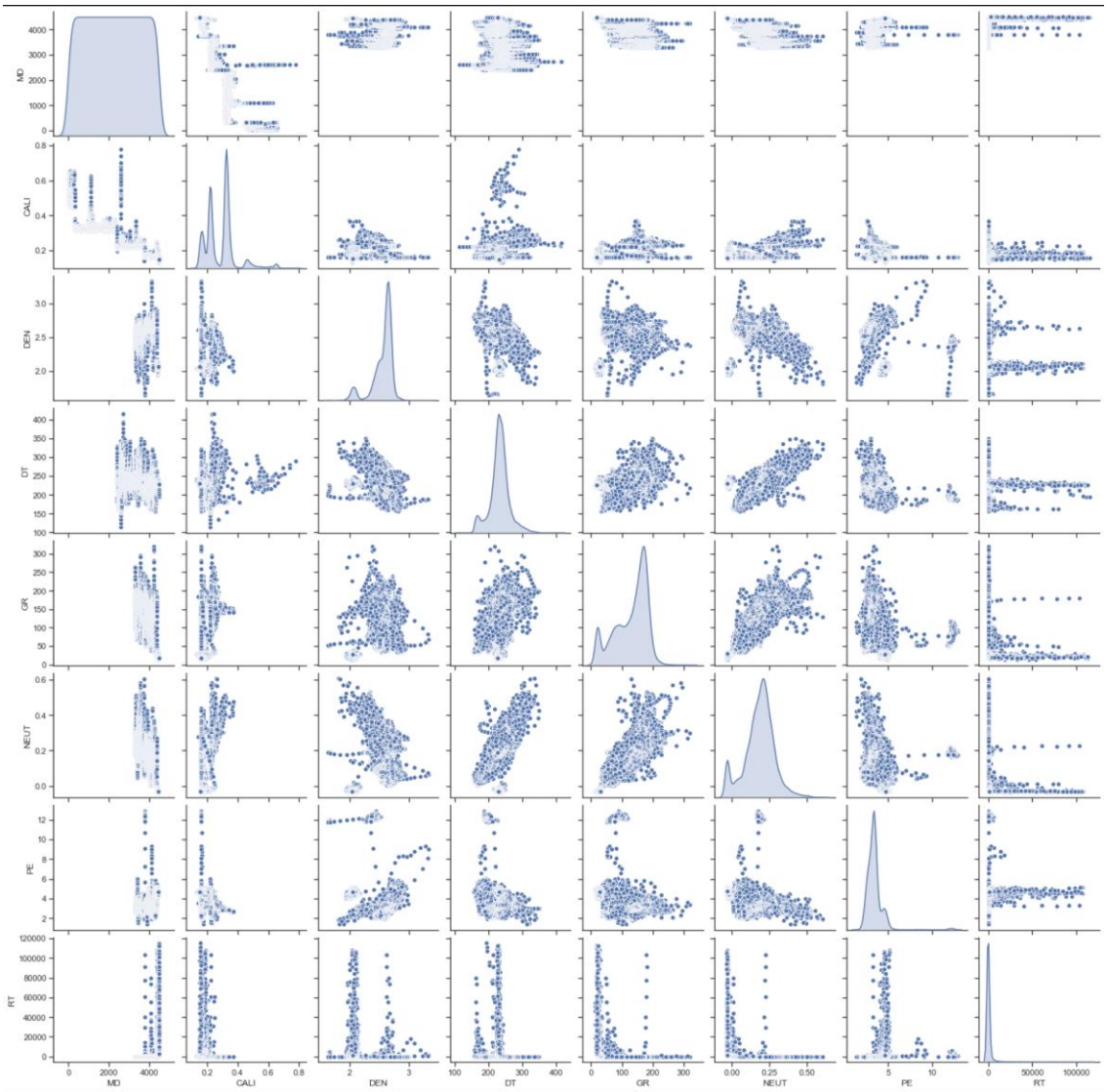


Рисунок 3.1 – Вхідні дані у вигляді skater-matrix

З огляду на скатер-матрицю (рис. 3.1) можна попередньо зробити висновок, шо в основному, якщо грубо оцінити, дані представляють собою один великий кластер. Можна на перед сказати, що результати методу K-Means будуть поганими адже точки доволі тісно скупчені в одній область і центри кластерів будуть в основному вираховуватись відносно параметру вхідних данних – RT, завдяки його сильній дисперсії.

А ось вхідні данні представлені у вигляді графіків, наближених до вигляду справжніх каротажів гірничодобувної промисловості (до даних додано незначний шум, який не впливає на графоаналітичну оцінку людиною, бо данні на цей проект були надані за умовою, що вони не потраплять у відкритий доступ) як показано на рисунку 3.2:

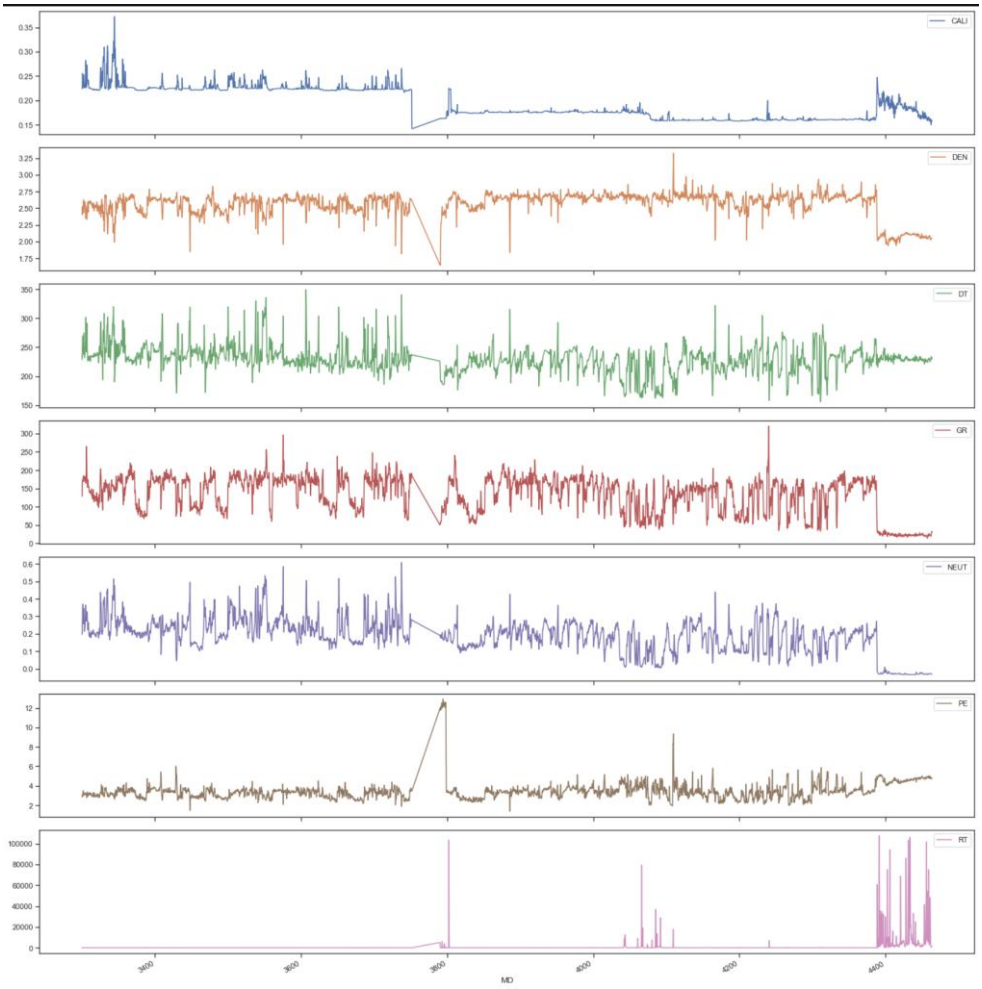


Рисунок 3.2 – Каротажі вхідних даних, які наближені до вигляду справжніх каротажів гірничодобувної промисловості

Отже значення вхідного набору даних набувають таких меж, де вісь абсцис – значення вхідного набору даних яке відповідає за прокладену відстань від поверхні землі до фінальної точки роботи під землею (MD):

– CALI (0.15 - 0.35 m) – фактичний розмір поперечного перетину – діаметр утворений під час проходу буру;

– DEN (1.75 – 3.25 g/cm^3) – сума щільності породної системи;

– DT (150 – 350 us/m) – сума акустичних хвиль (невідомі прилади та їх налаштування);

– GR (0 – 300 $gAPI$) – результат бомбардування гамма частками;

– NEUT (0 – 0.6 v/v) – компенсована пористість нейтронів;

– PE (2 – 12 $b/elec$) – фотоелектричний ефект;

– RT (0 – 100000 $Ohmm$) – питомий опір.

Неозброєним оком можна побачити на рисунку 3.2, що параметр RT – набуває чітко виражених аномальних значень, що вносить не бажаний шум який спричинятиме не коректний результат кластеризації для багатьох методів алгоритм яких базується на підрахунках відносно центрів кластерів, такі як наприклад K-Means. Водночас цей самий параметр вздовж більшості шляху по осі абсцис взагалі не змінюється, що означає некоректні результати кластеризації для алгоритмів які базуються на щільнісній характеристиці – спричинятиме «мости», які будуть об’єднувати кластери які не мали б об’єднатись як показано на рисунку 3.3:

Рисунок 3.3 – Кореляційна матриця вхідних даних

На рис. 3.3 представлена кореляційна матриця, де якомога світлий колір блоку означає сильну пряму кореляцію параметрів (наприклад по діагоналі – значення кореляції параметра самого з собою, що очевидно, дорівнює одиниці), що знаходяться на перетині, а темний колір – слабку обернену кореляцію. Наприклад параметр NEUT дуже добре корелює з параметром GR, а CALI і PE навпроти як показано на рисунку 3.4:

	MD	CALI	DEN	DT	GR	NEUT	PE	RT
count	11258.000000	11258.000000	11258.000000	11258.000000	11258.000000	11258.000000	11258.000000	11258.000000
mean	3885.656529	0.193254	2.563709	229.058114	132.924748	0.185637	3.462852	994.528764
std	341.004682	0.031093	0.174837	22.760733	50.156707	0.097486	1.006111	6698.835679
min	3300.000000	0.141841	1.643747	156.104592	13.598228	-0.033976	1.389932	0.000629
25%	3581.425000	0.161935	2.506154	217.973867	96.907234	0.136537	2.966422	8.313573
50%	3900.950000	0.177231	2.623704	228.817605	148.511831	0.196003	3.364418	13.234294
75%	4182.375000	0.223297	2.670689	240.046494	171.892765	0.242695	3.672650	24.858967
max	4463.800000	0.372137	3.327498	349.351136	320.621915	0.607464	12.911657	107726.990800

Рисунок 3.4 – Таблиця опис вхідних даних

На рис. 3.4 зображено, у вигляді компактної таблиці, основні характеристики вхідного набору даних по параметрам:

– count – кількість змінних;
– mean – арифметичне середнє значення;
– std – стандартне відхилення (корінь від дисперсії);
– min – мінімальне значення;
– 25% - двадцять п’ятий персентиль (0.25 квантиль);
– 50% - п’ятдесятий персентиль (0.5 квантиль);
– 75% - сімдесят п’ятий персентиль (0.75 квантиль);
– max – максимальне значення.

Як бачимо, параметри – 7 змінних значень (точки за якими і буде проводитись кластеризація), та змінна MD (яка виступає в ролі часового ряду вздовж якого і відбувається по суті аналіз) приймають різну область значень. Тому дані потрібно нормалізувати, адже, більшість методів кластеризації працюють саме з нормалізованими даними. Наприклад, метод K-Means – можна представити його роботу у вигляді набору сфер, тобто якжо точка потрапляє в сферу то ця точка належить цьому кластеру, якщо не нормалізувати данні то завдяки його алгоритму побудованому на фактичній відстані до точок і центрів кластерів, він проведе кластеризацію по параметру в якого найсильніша дисперсія («сфери» будуть над-то великі і будуть не враховувати відстані по іншим вимірам/параметрами) – параметр RT. Згідно до інформації з розділу №2, є припущення що високим значенням

RT відповідають області з підвищеною вологістю, вкрапленням рудних мінералів, або навіть області протікання ґрунтових вод.

3.3 Нормалізація і кластеризація

З допомогою правильно підібраного способу кластеризації можна не тільки привести параметри до однакової області визначення, а також явно вирізнити аномальні значення, які утворюють «викиди» (в термінах кластеризації викид – значення які дуже контрастують по відношенню до основної маси елементів вибірки і в методах основаних на характеристиці відстані можуть спотворити результат кластеризації).

Нормалізація проводилась за формулою:

$$x_{ij\ norm} = \frac{x_{ij} - a_j}{b_j - a_j} i = \overline{1,n}, j = \overline{1,m} \#(3.1)$$

де n –кількість елементів в вхідному наборі даних;

m – кількість параметрів;

a_j і b_j – квантелі, рівно віддалені від 0.5 квантеля. Дивлячись на рисунок 3.4 було вирішено в даній роботі розглядати два варіанти нормалізації даних за 0.05 і 0.95 квантелями, та 0.1 і 0.9 квантелями. Такі нормалізації мають привести область визначення всіх параметрів до інтервалу – (0,…,1) та не враховувати в нормалізації аномальні значення параметрів – ті які створюють над-то велику дисперсію. Далі відбуваються довгі за часом підрахунки за одинадцятьма різними методами з більш ніж 4700 комбінаціями різних гіпер-параметрів відповідно до методу кластеризації. Загальний час підрахунків за всіма методами складав ~ 111 годин.

3.4 Оцінка результатів кластеризацій.

Результатом роботи являється більш ніж 4700 наборів різних міток. Таку кількість дуже складно переглянути людині та об’єктивно оцінити кожен результат кластеризації, навіть якщо представити їх у вигляді зручних скатер-матриць (які будуть наведені на рисунках нижче). Тому в якості оцінок якості кластеризації було застосовано такі метрики:

- Індекс Calinski–Harabasz;
- Індекс Davies-Bouldin;
- Коефіцієнт Silhouette.

Але якщо надати користувачеві лише таблицю розмірності n на 3, де n – кількість методів та їх конфігурацій, а число 3 – кількості метрик оцінювання, то це аж ніяк не допоможе швидко визначити зразки хорошого результату кластеризації, це навіть збільшить час для аналізу людиною. Тому, маючи ці три оцінки для кожного набору міток можна відсортувати (вирізнити) за спаданням результати які мають найкращі оцінки, але вважаючи що кожна оцінка має однаковий рівень значущості ми повинні одночасно вирізняти за трьома оцінками. Саме на цьому етапі застосовується критерій оптимальності по Парето. Для наочності і зменшеню обсягу будуть розглядатися перші оптимальні по Парето результати методів, які продемонстровані на рисунку 3.5:

Drop (without MD) norm by quantile (0.05-0.95)

	index	Calinski and Harabasz Score	Davies-Bouldin Score	Silhouette Score	method
0	AgglomerativeClustering(affinity='euclidean', ...	-0.898716	-0.036569	-0.927308	AgglomerativeClustering
3	Birch(branching_factor=20, compute_labels=True...	-0.887383	-0.041189	-0.908103	Birch
47	DBSCAN(algorithm='brute', eps=1.0, leaf_size=3...	-0.167986	-0.144841	-0.926210	DBSCAN
55	KMeans(algorithm='auto', copy_x=True, init='k-...	-0.980038	-0.063986	-0.740340	KMeans
61	MiniBatchKMeans(batch_size=50, compute_labels=...	-0.683674	-0.121157	-0.383701	MiniBatchKMeans
120	MeanShift(bandwidth=3, bin_seeding=False, clus...	-0.479903	-0.040646	-0.940837	MeanShift
123	OPTICS(algorithm='auto', cluster_method='dbsca...	-0.172512	-0.241855	-0.800214	OPTICS
127	SpectralClustering(affinity='nearest_neighbors...	-0.416187	-0.162208	-0.491898	SpectralClustering
129	SpectralBiclustering(init='k-means++', method=...	-0.909889	-0.042251	-0.881778	SpectralBiclustering

Drop (without MD) norm by quantile (0.1-0.9)

	index	Calinski and Harabasz Score	Davies-Bouldin Score	Silhouette Score	method
0	AgglomerativeClustering(affinity='euclidean', ...	-0.092692	-0.000000	-1.000000	AgglomerativeClustering
5	Birch(branching_factor=30, compute_labels=True...	-0.356042	-0.010031	-0.994901	Birch
68	KMeans(algorithm='full', copy_x=True, init='k-...	-0.401886	-0.012698	-0.981430	KMeans
76	MiniBatchKMeans(batch_size=50, compute_labels=...	-0.386722	-0.014295	-0.979518	MiniBatchKMeans
168	OPTICS(algorithm='auto', cluster_method='dbsca...	-0.093247	-0.174942	-0.989584	OPTICS
171	SpectralBiclustering(init='k-means++', method=...	-0.233652	-0.019036	-0.960528	SpectralBiclustering

Рисунок 3.5 – Таблиця з методами та їх відємними нормалізованими оцінками, отриманих за критерієм оптимальносі Парето

Через те що, щоб рішення $x \in X$, було оптимальним по парето , необхідно і достатньо щоб воно було рішенням наступної задачі багатокритеріальної оптимізації:

$$\max_{x \in X} \sum_{i=1}^{n} \lambda_i H_i(x), \ \#(3.2)$$

де параметри $\lambda_i,\ i = \overline{1,n},$ задовольняють умовам $\sum_{i=1}^{n} \lambda_i = 1,\ \lambda_i \in (0,1)$. Вданному випадку рівнозначні, тому необхідно нормалізувати значення коефіцієнтів. Оскільки коефіцієнт Silhouette лежить в інтервалі від 0 до 1, а два інших індекси ні, то треба їх нормалізувати тільки їх за формулою:

$$H_{i\,norm}(x) = \frac{H_i(x) - \min H_i(x)}{\max H_i(x) - \min H_i(x)} \quad \#(3.3)$$

і маємо задачу багатокритеріальної оптимізації з новими нормалізованими оцінками (рис 3.6 – 3.7):

$$\max_{x \in X} \sum_{i=1}^{n} \lambda_i H_{i\,norm}(x) \quad \#(3.4)$$

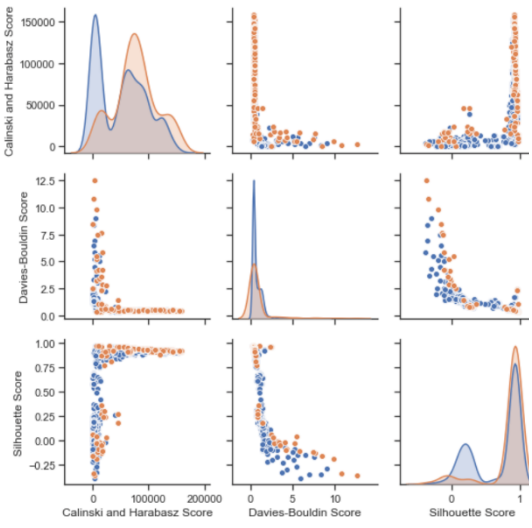


Рисунок 3.6 Графік - результат відбору методів за критерієм Парето (нормалізація за 0.1 і 0.9 квантелями)

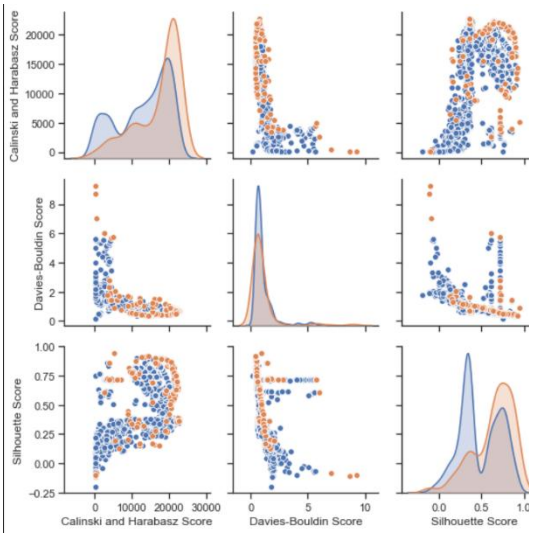


Рисунок 3.7 Графік - результат відбору методів за критерієм Парето (нормалізація за 0.05 і 0.95 квантелями)

На рис. 3.6 і рис. 3.7 зображені оцінки для двух варіантів нормалізації даних, де точки оранжевого ідентифіковані як оптимальні по Парето. Всі інші точки (синього кольору) – результати кластеризації можна відкинути.

3.5 Результати кластеризацій.

Ось приклад непоганого результату кластеризації представлений у вигляді скатер-матриці як показано на рисунку 3.7, алгоритм SpectralCluctering:

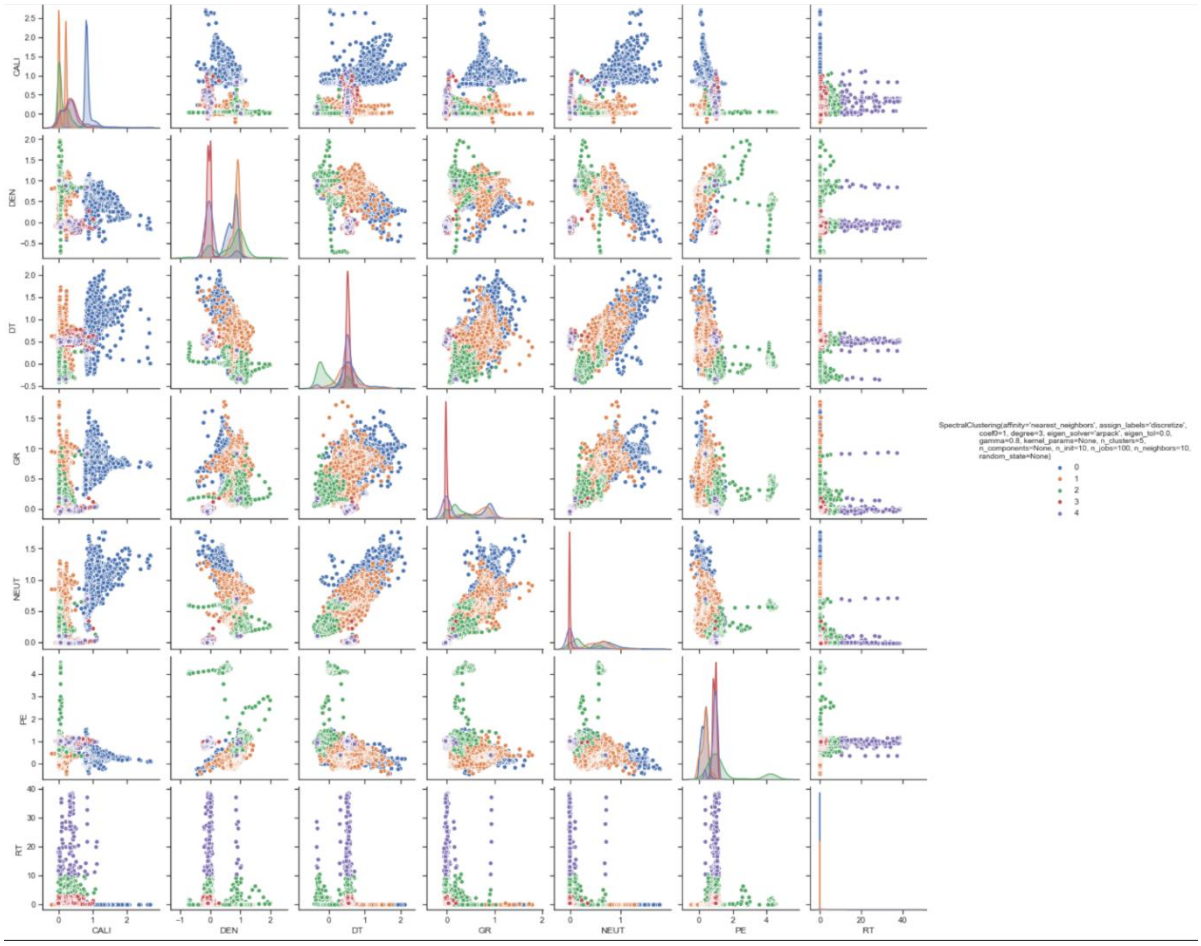


Рисунок 3.7 – Результат кластеризації представлений у вигляді скатер-матриці, алгоритм SpectralCluctering

Кожен блочний графік, крім тих що на головній діагоналі, відповідають 2-вимірному відображенню точок де індекс параметру зліва по вертикалі

відповідає за вісь ординат для кожного блок-графіка, а індекс параметру знизу по горизонталі відповідає за вісь абсцис для кожного блок-графіка відповідно як показано на рисунку 3.8:

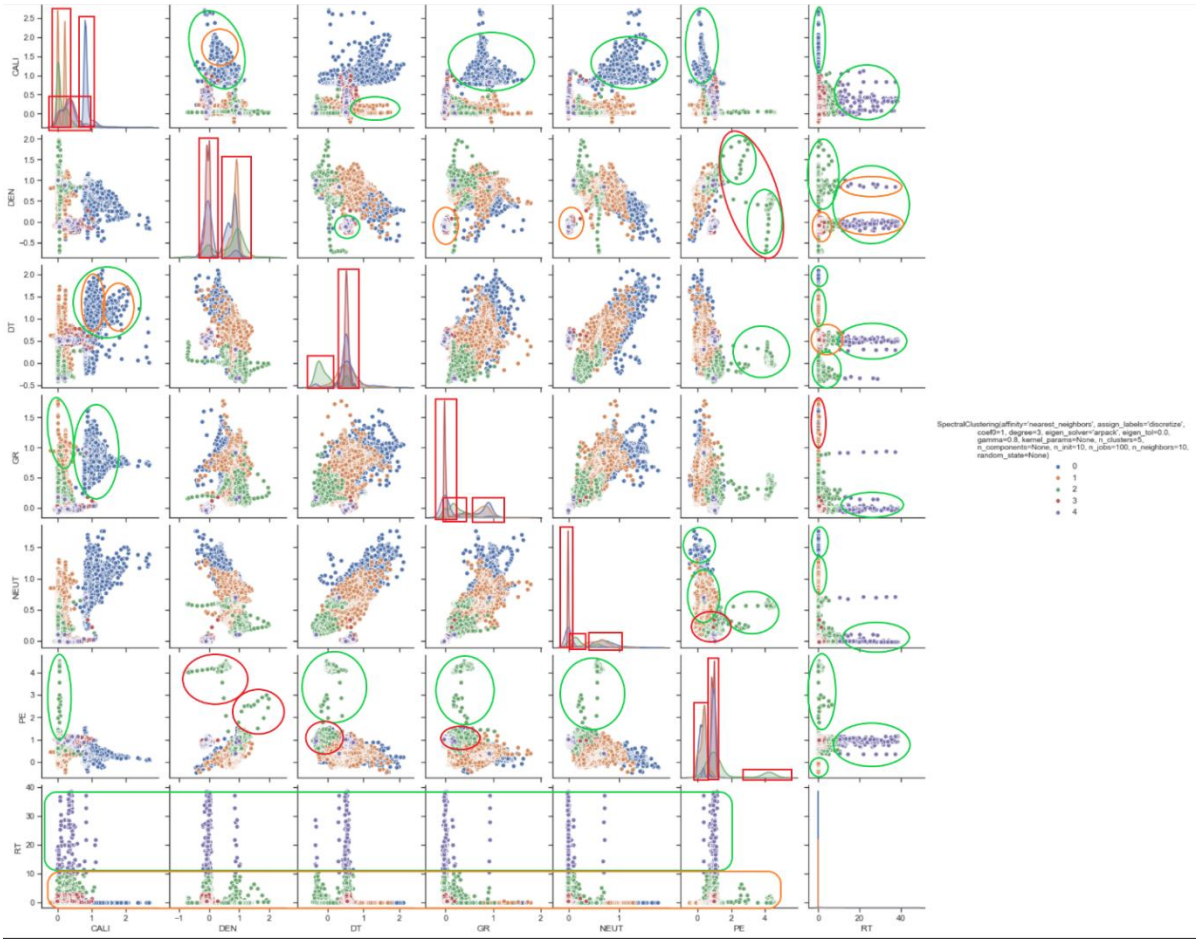


Рисунок 3.8 – Проаналізовани результат кластеризації представлений у вигляді скатер-матриці, алгоритм SpectralCluctering

На головній лінії (рис. 3.8) розташовані діаграми – оцінки щільності ядр кластеризації, чим вище «шпиль» тим більше точок розміщено на проміжку, яким відповідає колір, на відрізку осі абсцис, а для осі ординат – таж сама логіка але за шириною «шпиля».

– Червоний прямокутник на головній діагоналі – явні кластери за щільністю (може перекриватися візуально на блок-графіках);

– Зелена окружність – приклад гарного виявлення кластеру;

– Помаранчева окружність – бажана область до віднесення до нового кластеру;

– Червона окружність – приклад поганої кластеризації.

Проаналізувавши скатер-матрицю можна прийти до висновку що за багатьма вимірами червоний кластер повністю накриває фіолетовий, це означає що це в принципі один і той самий кластер, але, як і зазначалось раніше, параметр RT очікувано спричинив сильні «викиди», за якими і був ідентифікований новий – фіолетовий кластер.

Проаналізувавши за обома варіантам нормалізованих даних всі представлені Парето оптимальні результати, було визначено що найкращий варіант кластеризації був проведений за методом OPTICS за нормалізацією (0.05-0.95), з такими параметрами OPTICS(algorithm='auto', cluster_method='dbscan', eps=0.5, max_eps=1, metric='cityblock', min_samples=15), де:

– algorithm – алгоритм за яким будуть вираховуватися найближчі сусіди;

– cluster_method – метод кластеризації;

– eps – максимальна відстань до точки, що розглядається, до іших точок, які будуть розглядатися як сусідні;

– max_eps – максимальна відстань навколо точки, в області якої слід шукати сусідів;

– metric – алгоритм за яким буде вираховуватися відстань від точки до точки (при бажанні можна використати навіть власний алгоритм вирахування відстані);

– min_samples – кількість сусідніх точок потрібна для віднесення точки до кластеру.

Чому саме цей результат був ідентифікований як найкращий – рисунок 3.9:

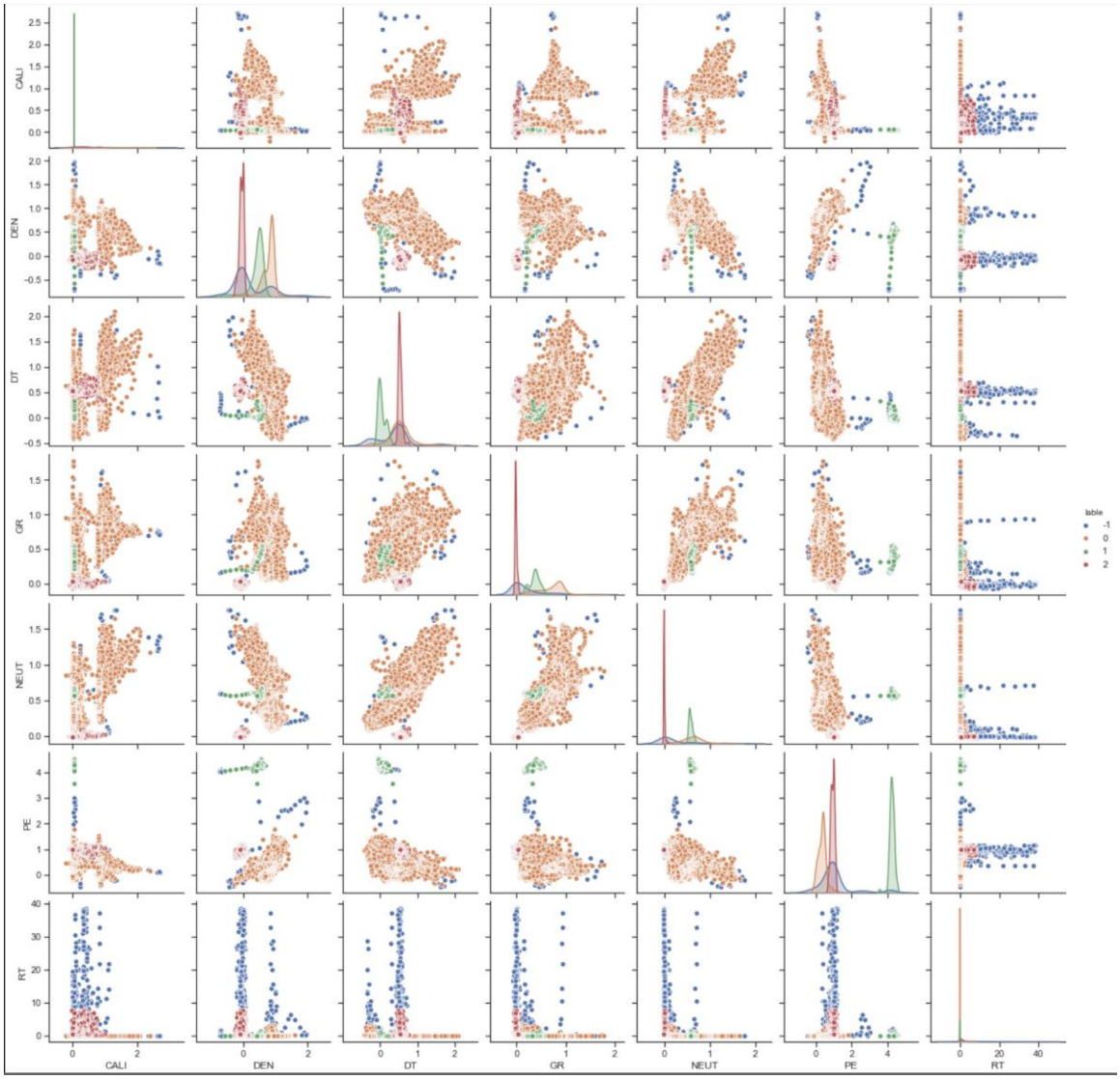


Рисунок 3.9 – Результат кластеризації представлений у вигляді скатер-матриці, алгоритм OPTICS

Оскільки метод є надбудовою над методом DBSCAN, то алгорит його базується на щільнісній характеристиці розподілу вхідних точок, що для цих даних дуже доречно. А безпосередньо сам метод має додаткову вагому можливість лімітувати область пошуку для одного кластеру – це дозволяє контролювати вплив небажаних «мостів» як показано на рисунку 3.10:

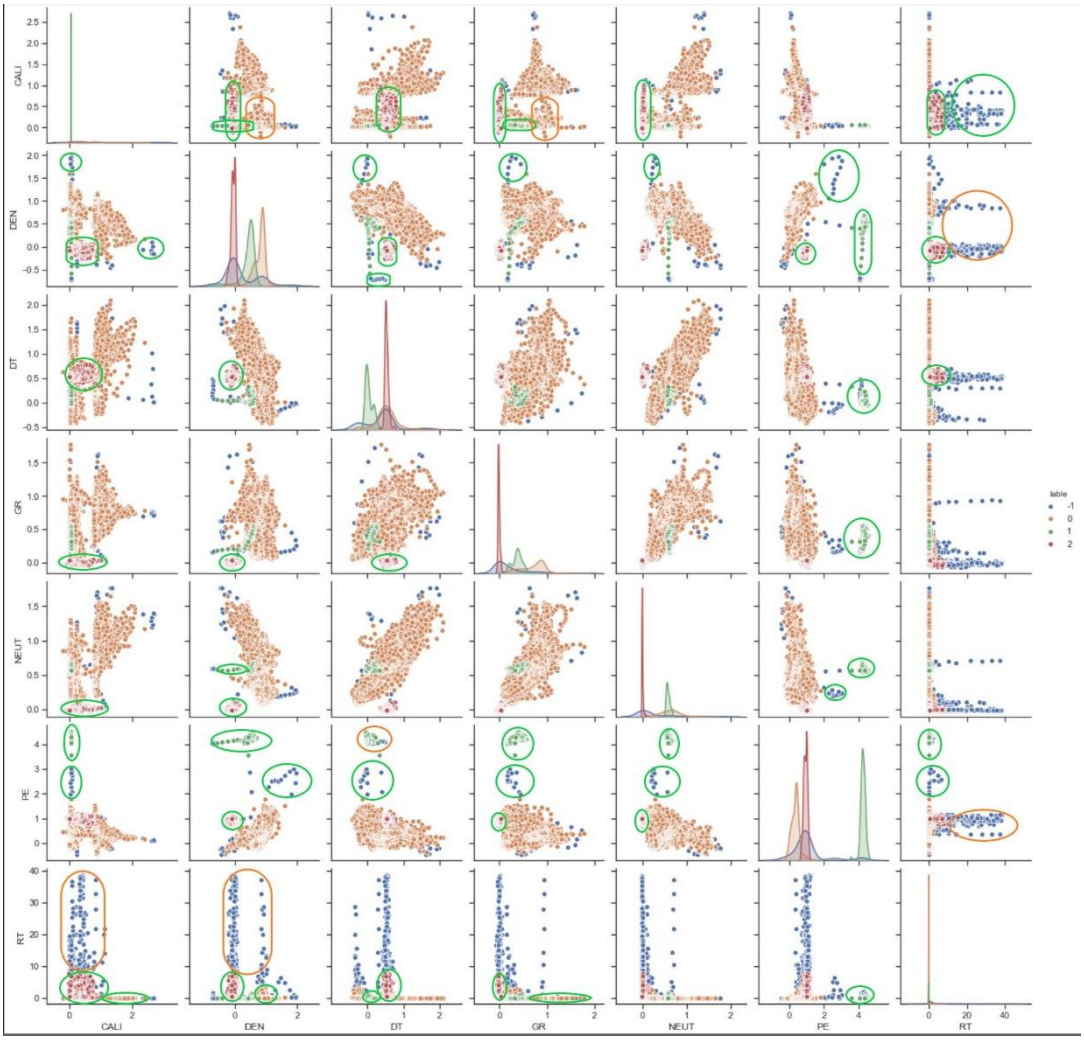


Рисунок 3.10 – Проаналізовани результат кластеризації представлений у вигляді скатер-матриці, алгоритм OPTICS

Було дуже не погано розмежовані області по проблемному параметру – RT. Занадто високі значення не були віднесені до жодного з кластерів (кластер з індексом «-1» - елементи що не потрапили до жодного з кластерів), через велику область значення та малу щільність точок, що знаходяться вище значення ~ 9 нормалізованих даних за 0.05 і 0.95 -ми квантелями. Як видно на рис. 3.9 – попри аномальну, на фоні інших параметрів, поведінку значенням RT не було знехтувано, як на результатах, зображення яких буде продемонстровано далі, і кластеризація за цим параметром відбулась доволі успішно (увагу на лінію з графіками під вертикальним індексом RT – рис. 3.10).

Кластери, окрім домінуючого – оранжевого кольору, інтуїтивно добре вирізняються на тому чи іншому графіку (рис. 3.9 – 3.10). Самі ж ці кластери

незважаючи на непрості форми – далекі від сферичної форми, добре виражені. На графіках щільностей, що на головній діагоналі рис. 3.9 – 3.10), діаграми відносно інших результатів, мінімально перетинаються як показано на рисунку 3.11:

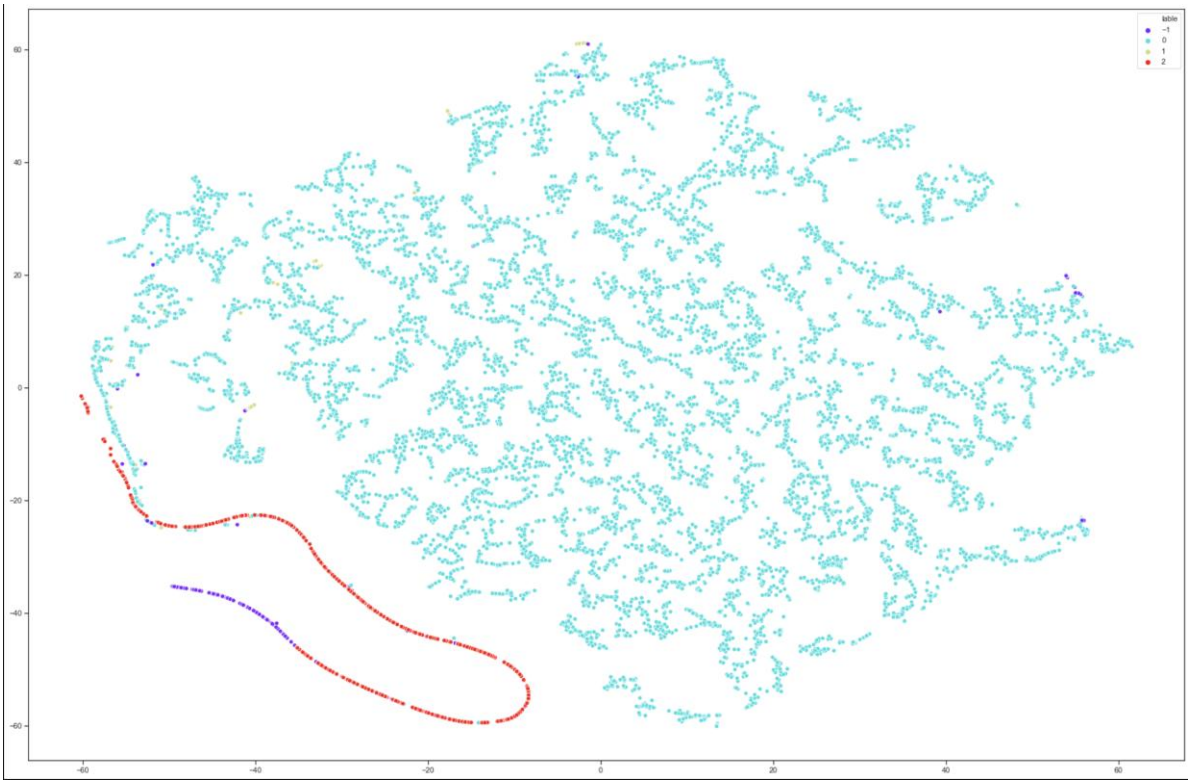


Рисунок 3.11 – Розмічені (методом OPTICS) вхідні дані зменшеної розмірності методом TSNE

Для наочності, на рис. 3.11 зображені вхідні дані розмірність яких була зменшено до двовимірного простору, методом пониження розмірності TSNE? де точки:

– бірюзового кольору – кластер з індексом «0»;

– світло-коричневий – кластер з індексом «1»;

– червоний – кластер з індексом «2»;

– ффіолетовим – кластер з індексом «-1» (точки що не були віднесені до жодного з кластерів).

Результатом TSNE є данні зменшеної розмірності. Дуже гарною властивістю цього метода є те, що він враховує і намагається зберегти структуру даних. З огляду на рис. 3.11 можа прийти до думки що такі дані

можна дуже добре кластеризувати (неозброєним оком можна навіть провести доволі непогану кластеризацію), але за результуючими дани цього методу не можна проводити кластерний аналіз - TSNE не зберігає точної відносної відстані та щільності, він більше бере акцент на збереження ієрархії між сусідніми точками як показано на рисунках 3.11(а) – 3.11(б):

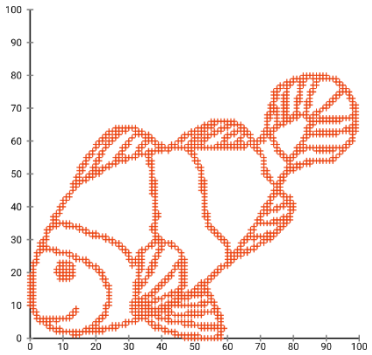


Рисунок 3.11(а) – Приклад - вхідні дані для TSNE

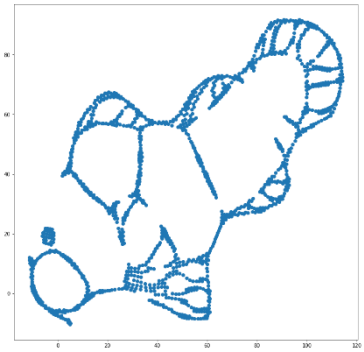


Рисунок 3.11(б) – Приклад - вихідні дані для TSNE

Основним результатом кластеризації є центри кластерів. Завдяки правильно підібраному методу кластеризації з його параметрами включно можна отримати правильно, відносно поставленої задачі і області дослідження, розмічені кластери і як наслідок центри кластерів. За якими можна, вже без навчання моделі, відносити точки (до якого центру кластеру вхідна точка найближче, тому кластеру і належить) з нових результатів геодосліджень як показано на рисунку 3.12:

	CALI	DEN	DT	GR	NEUT	PE	RT
-1	0.189984	2.235100	218.487549	57.781666	0.062674	4.972510	35490.766335
0	0.194080	2.602232	229.466117	141.515131	0.201233	3.299244	45.014042
1	0.162926	2.372065	192.933880	87.145291	0.182964	12.235160	289.092297
2	0.184291	2.067170	230.149204	23.194469	-0.027436	4.622646	4719.441183

Рисунок 3.12 – Центри кластерів алгоритму OPTICS

Подальша конвертація індексів в справжні назви корисних копалин, характеристики яких, з деякою похибкою (залежить від якості кластеризації, що в свою чергу дуже залежить від розміру і якості вхідного набору), співпадатимуть зі значеннями центрів кластерів унеможливлюється через те, що:

– невідомий хоча б регіон де проводились заміри;

– невідомий тип, виробник, та фізичні характеристики устаткування;

– невідомі налаштування сенсорів, серед яких є дуже чутливі (найменша відмінність від попередніх налаштувань яких – може призвести до кардинально інших результатів замірювань).

Але не зважаючи на ці недоліки, людина – експерт у цій області, яка перебуває безпосередньо під час проведення цих замірів і знає що за устаткування і як вони налаштовані, зможе легко ідентифікувати за який тип корисних копалин відповідає той чи інший кластер.

3.6 Висновки до розділу 3

Сучасні методи кластеризації в області поверхневого аналізу геофізичних досліджень є дуже перспективними. Добре навчена модель може допомогти швидко проводити аналіз великих наборів даних, однакових за набором характеристик - параметрів.

Аналізуючи цю роботу що найкращі результати кластеризації, для розглядуваних в цій роботі даних, демонструють методи алгоритм яких базується на щільнісній характеристиці вхідного набору, а також мають властивість добре опрацьовувати «шуми». Наприклад DBSCAN та його по суті модифікація – OPTICS.

Через те що актуальність даних існує лише в доволі короткому проміжку часу (на це впливають погодні умови) та через те що прилади і методи замірів геофізичних досліджень не є уніфікованою – швидкість навчання на актуальних даних є доволі вагомою характеристикою моделі, для поверхневого аналізу. Це дозволить не застосовувати додаткові ресурсу (в разі явної відсутності підземних ресурсів , за для здобутку яких і проводять дослідження), для більш детального і точного аналізу, тим самим зекономив гроші та час на ділянках землі, що не мають перспектив для здобутку шуканих корисних копалин.

Недоліком цієї роботи являється те, що для фінальної ідентифікації кластерів як ту чи іншу корисну копалину – потрібна експертна, в цій галузі, людина що зможе провести ідентифікацію зважаючи на характеристику і налаштування інструменту, яким проводять підземний аналіз.

РОЗДІЛ 4ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ

4.1 Вступна частина

В даному розділі буде прорахований функціонально-вартісний аналіз розробки програмного продукту. Дана розробка реалізована на мові програмування Python у середовищі JupyterLab в якості модуля, який можна вбудовувати у веб-застосунок. Програмний продукт призначено для використання в першу чергу на персональних комп'ютерах, але також не виключається можливість використання на мобільних засобах комунікації.

4.2 Постановка завдання техніко-економічного дослідження

Розроблювальний програмний продукт (ПП) – система класифікації гірських порід та корисних копалин на базі показників роботи гірничо прохідної техніки за допомогою найбільш відомих методів класифікації.

Для забезпечення конкурентної спроможності організації, необхідно мати зручний і надійний інструмент, який зможе автоматизувати роботу у сфері діяльності організації, а не застосовувати людський ресурс – кваліфікованих експертів в цій галузі, яких важко віднайти та їх послуги коштують значно більше ніж очікувані витрати на розробку та підтримку програмного засобу, також це усуває людський фактор необачності. Ціллю цієї розробки було створити програму з оптимальним набором необхідних функцій і простим та зручним інтерфейсом, що не потребував би великої кількості часу на ознайомлення керуючого персоналу.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональнихкомп'ютерахізстандартним набором компонент;

– забезпечувативисокушвидкістьобробки великих об’ємівданих у реальному часі;

– забезпечуватизручність і простоту взаємодії з користувачемабо з розробникомпрограмногозабезпечення у випадкувикористовуванняйого як модуля;

– передбачатимінімальнівитрати на впровадженняпрограмного продукту;

4.3 Виділення основних функційаратетрів ПП

Головна функція – $F0$, означає розробку ПП, який дає найменшу похибку при порівняння реальних значень цільової змінної та результуючого значення побудованого прогнозу, у відповідній точці. Виходячи з конкретнихцілей, реалізованихпрограмнимзасобом, виділимоїїосновніфункції:

F_1 −вибірмовипрограмування; a) мовапрограмуванняPython;

b) мовапрограмування C++.

F_2 − використання готових бібліотек;a) використанняготовоїбібліотеки; b) написанняалгоритмівроботи з данимивручну.

a)F_3 − середовище розробки a) .IDE Pycharm; b) Jupiter lab.

Варіанти реалізації основних функцій наведено у морфологічній карті системи (рис. 4.1). На основіцієїкартипобудовано позитивно-негативнуматрицюваріантівосновнихфункцій (таблиця 4.1).

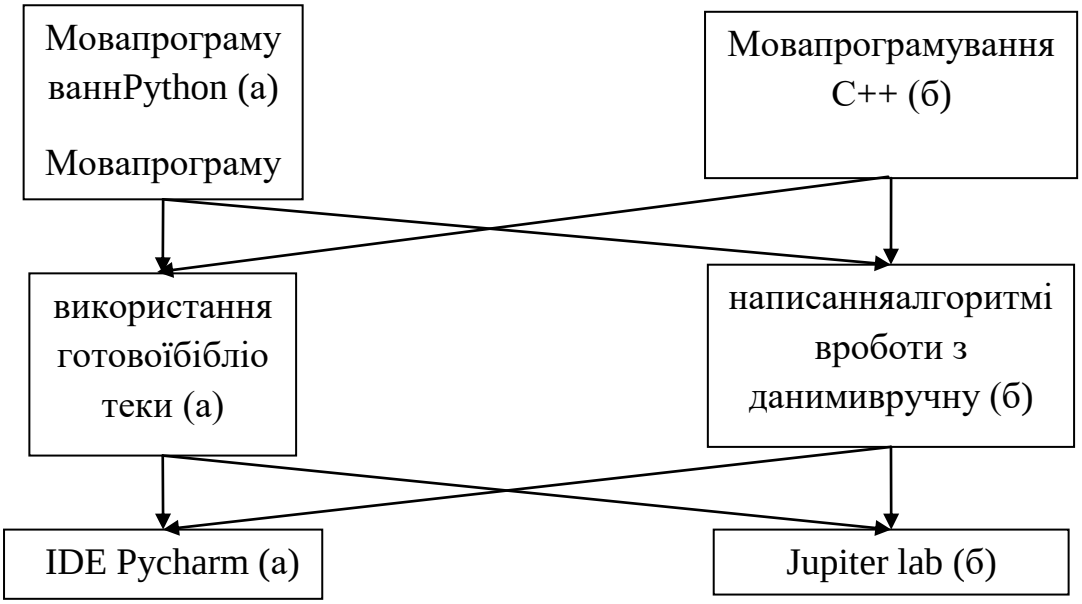


Рисунок 4.1Морфологічна карта

Морфологічна карта відображаєвсіможливікомбінаціїваріантівреалізаціїфункцій, якіскладаютьповнумножинуваріантівпрограмного продукту.

Основні функції	Варіанти реалізації	Переваги	Недоліки
F_1	А	Займає менше часу при написанні коду, кросплатформений	Динамічна типізація
	Б	Код швидко виконується, кросплатформений	Займає більше часу при написанні коду
F_2	А	Легкістьреалізації,економія часу	Меньша гнучкість
	Б	Оптимально для власних продуктів, блочна реалізація	Затрачений час, можливі помилки в синтаксисі
F_3	А	Присутня можливість відладки коду	Необхідна додаткова

			інсталяція
	Б	Широкий вибір можливостей	Відсутність відладки коду

Таблиця 4.1- Позитивно-негативна матриця

На основіаналізу позитивно-негативноїматриціробимовисновок, що при розробціпрограмного продукту деяківаріантиреалізаціїфункційвартовідкинути, тому, що вони не відповідаютьпоставленим перед програмним продуктом задачам.

Таким чином, будеморозглядатитаківаріантиреалізації ПП:

1) $F1а - F2а - F3а$

2) $F1а - F2а - F3б$

Для оцінюванняякостірозглянутихфункцій обрана система параметрів, описана нижче.

4.4 Обґрунтуваннясистемипараметрівпрограмного продукту

4.4.1 Описпараметрів

На підставіданих про основніфункції, що повинен реалізуватипрограмний продукт, вимоги до нього, визначаютьсяосновніпараметривиробу, щобудутьвикористані для розрахункукоефіцієнтатехнічногорівня.

Гірші, середні і кращізначенняпараметріввибираються на основівимогзамовника й умов, щохарактеризуютьексплуатаціюпрограмного продукту. Інформація наведена у табл. 4.2.

Таблиця 4.2Система параметрівдодатку

Найменування параметру	Позначення параметру	Значення параметру		
		Гірші	Середнє	Кращі
Швидкодіямовипрограмування, с	X1	22	10	6
Об’ємпам’яті для збереженняданих , кб	X2	360	240	100
Час навчаннямоделі, хв	X3	400	300	200

За данимитаблиці 4.2будуютьсяграфічні характеристики параметрів – (рис. 4.1 – рис. 4.3).

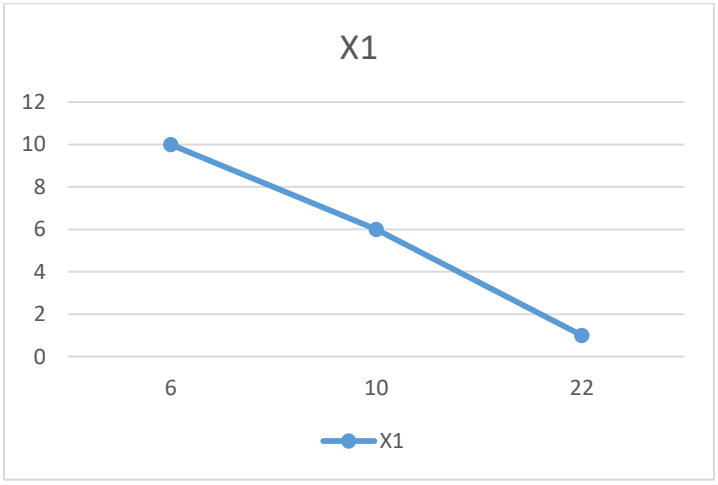


Рисунок 4.1 – X1 швидкодія мови програмування

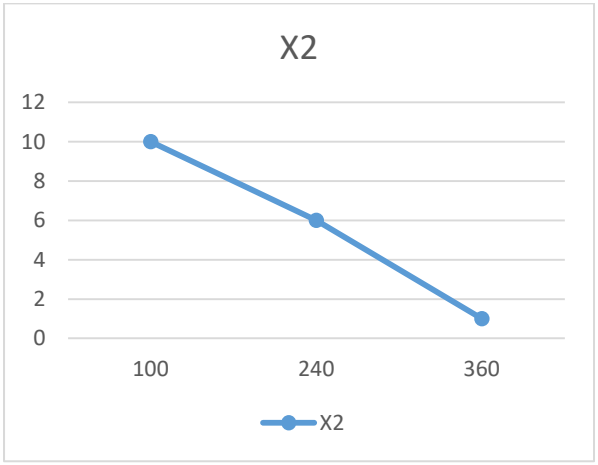


Рисунок 4.2– X2, об'ємпам'яті для збереженняданих

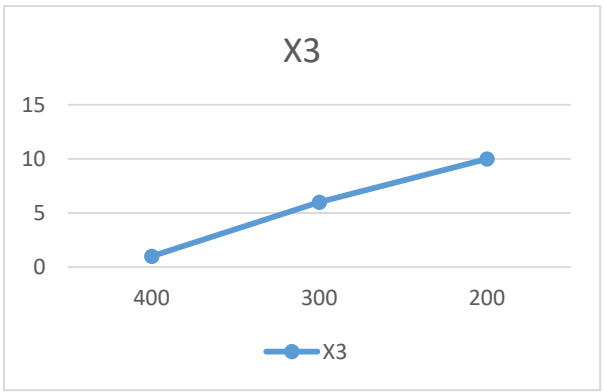


Рисунок 4.3 – X3, час обробкиданих алгоритмом

Результатиекспертногоранжуваннянаведені у таблиці 4.3

Параметр	Ранг параметру по оцінціексперта							Сума рангів, R_i	ВідхиленняΔ_i	Квадрат відхилення, $(\Delta_i)^2$
	1	2	3	4	5	6	7			
X1	3	3	3	3	2	3	3	20	6	36
X2	1	1	2	1	1	1	1	8	-6	36
X3	2	2	1	2	3	2	2	14	0	0
Разом	6	6	6	6	6	6	6	42	0	72

Найменший ранг – 1, найбільший ранг – 3.

Таблиця 4.3 Результатиранжуванняпараметрів:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 * 72}{49 * (27 - 3)} = 0{,}735 > W_k = 0{,}67$$

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.Найменший ранг – 1, найбільший ранг – 3.

Параметри	**Експерти**							**Кінцеваоцінка**	**Числове значення**
X1 та X2	>	>	>	>	>	>	>	>	1.5
X1 та X3	>	>	>	>	<	>	>	<	0.5
X2 та X3	<	<	>	<	<	<	<	<	0.5

Розрахуноквагомостіпараметрів наведено в табл. (4.5)

Як видно з таблиці (4.5), різницязначенькоефіцієнтіввагомості не перевищує 2%, тому більшої кількостііітерацій не потрібно.

Параметри	ПараметриX_i			Перший крок		Другийкрок	
	X1	X2	X3	b_i	K_{bi}	b_i	K_{bi}
X1	1	1,5	0,5	3	0,3333	8	0,32
X2	0,5	1	0,5	2	0,2222	5,5	0,22
X3	1,5	1,5	1	4	0,4444	11,5	0,46
Загалом:				9	1	25	1

Таблиця 4.5 Розрахуноквагомостіпараметрів

Залишаютьсянаступніваріанти:

1)F1(а) => F2(а) => F3(а)

2)F1(а) => F2(а) => F3(б)

Основна функція	Варіант реалізації	Абсолютнезначення параметру	Бальнаоцінка параметру	Коефіцієнтвагомості параметру	Коефіцієнтякості
F1	а)X1	10	6	0,32	1,92
F2	а)X2	240	6	0,22	1,32
F3	а)X3	340	4	0,46	1,84
	б)X3	250	8	0,46	3,68

Таблиця 4.6 Розрахунокпоказниківрівняякостіваріантівреалізації:

$$K_{я1} = 1{,}92 + 1{,}32 + 1{,}84 = 5{,}08$$

$$K_{я2} = 1{,}92 + 1{,}32 + 3{,}68 = 6{,}92$$

Оскількиваріант 2 маєнайбільшийкоефіцієнтякості, вінєнайкращим.

4.5 Економічний аналіз варіантів розробки

Для визначеннявартостірозробки ПП спочаткупроведеморозрахуноктрудомісткості.

Всіваріантивключають в себе два окремихзавдання:

1.Розробкапроекту програмного продукту;

2.Розробкапрограмної оболонки;

При цьому два варіанти мають різнедодатковезавдання:

3.1.Реалізація методів аналізу. Для першого варіанту.

3.2.Створення нового методу аналізу. Для другого варіанту.

Завдання 1 за ступенемновизнивідноситься до групи В, ступіньскладності 1трудомісткістьдорівнює: $Т_Р$ = 44людино-днівзавдання.Поправочнийкоефіцієнт, якийвраховує вид нормативно-довідковоїінформації для першогозавдання: $К_П$ = 0,77 Поправочнийкоефіцієнт, якийвраховуєскладність контролю вхідної та вихідноїінформації для всіхзавданьрівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.86. Тоді, трудомісткістьпрограмуванняпершогозавданнядорівнює:

$$Т_1 = 44 * 0,77 * 0,86 = 29,14 \text{ людино-днів}$$

Проведемоаналогічнірозрахунки для подальшихзавдань.

Для другого завданнявикористовується алгоритм другоїгрупискладності, степіньновизни Б, тобто

$Т_р = 25$ людино-днів, $К_п = 0,92$, $К_{ск} = 1$, $К_{ст} = 0,8$:

$$Т_2 = 25 \cdot 0,92 \cdot 0,8 = 18,4 \text{ людино-днів}$$

Для завдання три(а) використовується алгоритм третьоїгрупискладності, степіньновизниГ тобто:

$Т_р = 11$ людино-днів, $К_{ск} = 1$, $К_{ст} = 0{,}66$, $К_П = 0.58$

$Т_{р(а)} = 11 \cdot 0{,}66 \cdot 0.58 = 4{,}21$ людино-днів

Для завдання три(Б) використовується алгоритм першоїгрупискладності, степіньновизниБ,тобто:

$Т_р = 9$ людино-днів, $К_{ск} = 1$, $К_{ст} = 0{,}66$, $К_П = 0{,}33$

$Т_{р(б)} = 9 \cdot 0{,}66 \cdot 0{,}33 = 1{,}96$ людино-днів

Складаємотрудомісткістьвідповіднихзавдань для кожного з обранихваріантівреалізаціїпрограми, щоботриматиїхтрудомісткість:

$Т_I = (29{,}14 + 18{,}4 + 4{,}21) \cdot 8 = 414$ людино-годин

$Т_{II} = (29{,}14 + 18{,}4 + 1{,}96) \cdot 8 = 396$ людино-годин

Найбільшвисокутрудомісткістьмаєваріант II.

В розробціберуть участь один програміст з окладом 20000 грн., один дата аналітик з окладом 16000грн. Визначимо зарплату за годину за формулою:

$$С_ч = (20000 + 16000грн)/(2 * 25 * 8) = 90 \text{ грн}$$

$$С_{зп} = 90 \cdot 414 \cdot 1{,}2 = 44712 \text{ грн.}$$

$$С_{зп} = 90 \cdot 396 \cdot 1{,}2 = 42768 \text{ грн.}$$

Відрахування на соціальнийвнесок становить 22%:

$$С_{від} = С_{зп} \cdot 0.22 = 44712 \cdot 0.22 = 9836{,}64 \text{ грн.}$$
$$С_{від} = С_{зп} \cdot 0.22 = 42768 \cdot 0.22 = 9408{,}96 \text{ грн.}$$

Тепервизначимовитрати на оплату однієїмашино-години (СМ).

Так як одна ЕОМ обслуговує одного програміста з окладом 14000 грн., та датасєнтиста з окладом 18000 грн. з коефіцієнтомзанятості 0,2 то для однієї машиниотримаємо:

$$С_г = 12 * 20000 * 0{,}6 + 12 * 16000 * 0{,}6 = 259200$$

З урахуваннямдодаткової заробітної плати:

$$С_{зп} = С_г \cdot (1 + К_з) = 259200 \cdot (1 + 0.2) = 311040 \text{ грн.}$$

Відрахування на соціальний внесок 22%:

$$С_{ВІД} = 311040 * 0{,}22 = 68428{,}8 \text{ грн.}$$

Амортизаційнівідрахуваннярозраховуємо при амортизації 25% та вартості ЕОМ – 60000 грн.

$$С_А = К_{ТМ} * К_А * Ц_{ПР} = 1{,}15 * 0{,}25 * 60000 = 17250$$

Витрати на ремонт та профілактикурозраховуємо як:

$$С_Р = К_{ТМ} * Ц_{ПР} * К_Р = 1{,}15 * 60000 * 0{,}05 = 3450 \text{ грн.}$$

де $К_Р$ −відсотоквитрат на поточніремонти.

Ефективний годинний фонд часу ПК за рікрозраховуємо за формулою:

$$Т_{ЕФ} = (Д_К - Д_В - Д_С - Д_Р) * t_З * К_В,$$
$$Т_{ЕФ} = (365 - 142 - 16) * 8 * 0,8 = 1324,8 \text{ год}$$

Витрати на оплату електроенергіїрозраховуємо за формулою:

$$С_{ЕЛ} = 1324,8 * 0,6 * 0,6 * 1,75 = 834,624 \text{ грн.}$$

Накладнівитратирозраховуємо за формулою:

$$С_Н = 60000 * 0,67 = 40200 \text{ грн.}$$

Тоді, річніексплуатаційнівитратирозраховуємо за формулою:

$С_{ЕКС} = 276480 + 60825,6 + 17250 + 3450 + 834,624 + 40200 =$ 399040,224 грн.

Тодісобівартістьоднієїмашино-години ЕОМ дорівнюватиме:

$$С_{М-Г} = \frac{399040,224}{1324,8} = 301,21 \text{ грн/год}$$

Враховуючи, щовсіроботиведуться на ЕОМ, витрати на оплату машинного часу:

$$С_м = 301,21 * 414 = 124700,94$$
$$С_м = 301,21 * 396 = 119279,16$$

Накладнівитративідповідно:

$$С_Н = 44712 * 0,67 = 29957.04$$
$$С_Н = 42768 * 0,67 = 28654.56$$

Розрахуємоповнувартістьрозробки:

$$С_{ПП} = 44712 + 9836,64 + 124700,94 + 29957.04 = 209206,62$$
$$С_{ПП} = 42768 + 9408,96 + 119279,16 + 28654.56 = 200110,68$$

Розрахуємокоефіцієнттехніко-економічногорівня за формулою:

$$К_{ТЕР1} = \frac{5,08}{209206,62} = 24,28 * 10^{-6}$$
$$К_{ТЕР2} = \frac{6,92}{200110,68} = 34,58 * 10^{-6}$$

4.6 Висновки. Вибіркращоговаріанта ПП техніко-економічногорівня.

Отже враховуючи всі дослідження, що описані вище, можнасказати, що другий варіантреалізації є найбільш оптимальним зі сторони якісно-економічної оцінки. Його коефіцієнт техніко-економічного рівня складає $К_{ТЕР1}=34.58 * 10^{-6}$

Цейваріантреалізаціїпрограмного продукту маєтакіпараметри:

–мовапрограмування – Python;

–використанняготовихбібліотек;

–середовищерозробкиJupiter lab.

Даний варіантвиконанняпрограмного комплексу дає користувачу зручний інтерфейс, працюючі бібліотеки, широкий функціонал і швидкодію.

ВИСНОВКИ ПО РОБОТІ ТА РЕКОМЕНДАЦІЇ ЩОДО ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Основна мета роботи – поглибити знання в кластерному аналізі, петрофізиці і літології. Розробити програмний продукт на основі даних геологічних досліджень свердловини. Головними властивостями являються швидкодія програмного коду та масштабованість на різні технічні відмінності спорядження, що присутні у різних компаніях, якраз завдяки методам кластеризації, які дозволяють проводити аналіз непоганої якості на нерозмічених даних.

Складність полягає у дуже великій кількості «мостів» між даними, що погано впливає на чіткість кластерів, та «шуми» - які допомагають усунути методи які базуються на щільнісній характеристиці набору даних. В результаті дослідження було виявлено, що найбільш підходящі для цільової класифікаціх поставленої задачі являються методи DBSCAN і OPTICS. Також було досліджено метод зниження розмірності – TSNE, і придатність його результуючих даних до кластеризації.

В подальшій розробці планується біль точне використання методів кластеризації, поглиблення в підготовці та нормалізації даних.

ЛІІТЕРАТУРА

1. Білецький В. С. Мала гірнича енциклопедія в 3 т. Донецьк: Донбас, 2004. 640 с.
2. Dewan, J. T., Essentials of modern open-hole log interpretation. Tulsa: OK, Penn Well Books, 1983. 708 p.
3. Lecun Y., Boser B., Denker J. S. Atlas log interpretation charts. Houston, TX: Western Atlas International, 1985.
4. IEEE Conference on Computer Vision and Pattern Recognition. Seattle: Elliott, June 1994.
5. Karl Pearson, F.R.S. Mathematical contributions to the theory of evolution. Dulau: Co., 1904.
6. Calinski T., Harabasz J. A dendrite method for cluster analysis: Communications in Statistics, 1974.
7. Кузнецов В.Г. Литология: Осадочные горные породы и их изучение: Учебн. пособие для вузов, 2007. 511 с.
8. Tixier R.P., Alger C.A., Doh M.P. Sonic Logging. AIME, 1958
9. Biot M.A. Theory of Propagation of Elastic Waves in a Fluid-Saturated. America: Porous Solid, Journal of Acoustic Society, 1956.
10. Stripling A.A. Velocity Log Characteristics. Journal: Petroleum Technology, 1958.
11. Alberty, M. W., Morton-Thompson D., Woods A. M., Standard interpretation: wireline methods: part 4. Development Geology Reference Manual: AAPG Methods in Exploration Series 10, 1994 .180–185 p.
12. Зайченко Ю.П. Исследование операций: 2-изд. Киев: Изд-во «Вища школа», 1979.
13. Van der Maaten L.J.P., Hinton G.E. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research, 2008. 2579-2605.

14. Hans-Peter Kriegel, Peer Kröger, Jörg Sander, Arthur Zimek Density-based clustering: Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2011. 231–240.

15. Annual Report 2016 (SEC Filing Form 10-K). Schlumberger: ltd, 25 January 2017.

16. Онацький Є. Геологія: Українська мала енциклопедія : 16 кн. : у 8 т. Аргентина: Буенос-Айрес, 1958. 234 с.

ДОДАТОК А ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Інформаційно-аналітична система для визначення типів гірських порід за даними геофізичних досліджень свердловин

Виконав: Венгер Володимир Олегович
Науковий керівник: асистент кафедри ММСА
Макуха Михайло Павлович

Вступ

- В великих компаніях, що спеціалізуються на добутку корисних копалин, а саме: залізо, граніт, будівельний пісок, глина, сіль, гравій, гіпс, нафта, газ, золото тощо - аналіз ґрунтів проводиться методом «замірів високої чіткості».

Актуальність

- Заміри іноді проводяться на областях в декілька квадратних кілометрах - це є великий об'єм даних - матриця m на n (де m - кількість сенсорів різного застосування, n - відстань на яку було проведене буріння, заміри беруться кожні 10 сантиметрів - 0.1 метр) помножений на кількість замірів. Тому питання проведення поверхневого швидкго аналізу великих об'ємів вхідних даних є дуже актуальним для цих галузей.

Існуючі підходи для розв'язку

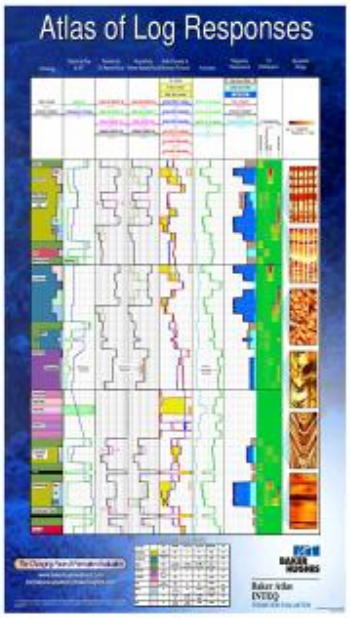
- Вивчення геологічного резерву свердловини за геофізичними даними - вивчення послідовності глибини залягання пластів гірських порід та їх літографічних властивостей, вміст корисних копалин і оцінки насиченості нафтою, газом або водою.
- Іноді для аналізу витягують так звані пластові ядра, але це не дуже корисно для аналізу адже вміст води, масла та газу змінюється а також тиск, щільність і тд.
- Найсучасніший спосіб - це опускати в свердловину скануючий зонд і робити заміри в режимі реального часу. Отримувати результати у вигляді каротажів і комбінувати їх.

Постановка задачі

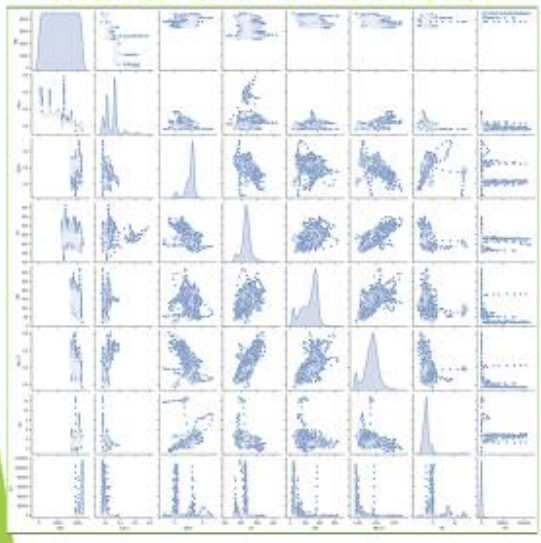
- Для цієї роботи було надано справжній результат сканування однієї з таких свердловин, але, нажаль результат лише одної і не було ніяких даних щодо типу устаткування, його виробника та налаштування, тому якість і результати кластеризації будуть залежати від якості заміру лише одного такого сканування, і можуть бути інтерпретовані спеціалістом предметної області. В чому даний програмний йому - експерту, допоможе зробити шляхом надання опрацьованих результатів в вигляді різноманітних графіків і матриць, для легкого і швидкого проведення поверхневої оцінки результатів заміру свердловин.

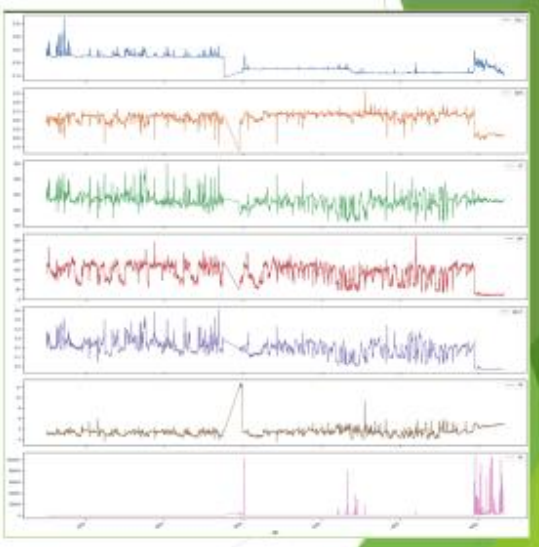
Мета

- Мета роботи - розробити алгоритм, який допоможе класифікувати корисні копалини, на тому чи іншому рівні під землею, з допомогою найбільш відомих методів кластеризації, реалізація яких присутня в бібліотеці мови програмування Python - scikit-learn, та інтерпретація даних у більш придатному для людського сприйняття вигляді, з допомогою методу зниження розмірності даних та графіків.

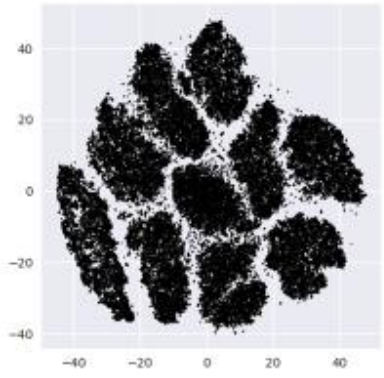


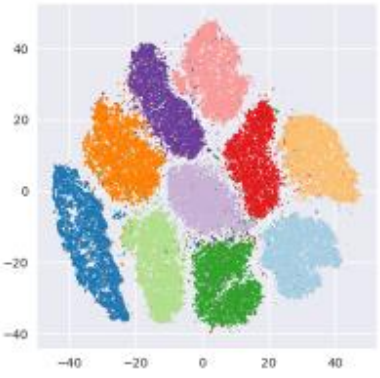
Математичні основи





Математичні основи



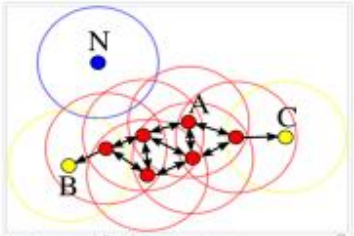


Метод DBSCAN (*density-based spatial clustering of applications with noise*)

Розглянемо множину точок в деякому просторі, яка будемо кластеризувати. Метою кластеризації процедурою DBSCAN є поділ точок на *ядрові*, *щільно-зв'язані* та *викиди*, які визначаються так:

- Точка p є ядровою, як хоча б minPts знаходяться на відстані ε (відстань ε є радіусом окола p) від неї, включно з p. Кажуть, що ці точки *безпосередньо досяжні* з p.
- Точка q безпосередньо досяжна з p, якщо точка q знаходиться на відстані не більшій ніж ε від точки ядрової точки p.
- Точка q є досяжною з p, якщо існує шлях $p_1, \ldots, p_n$ з точок $p_1 = p$ та $p_n = q$, де кожна p_{i+1} безпосередньо досяжна з p_i (всі точки шляху повинні бути ядровими, можливо за виключенням q).
- Всі точки не досяжні з будь-якої іншої точки є викидами.

Отже, ядрова точка p утворює «кластер» разом з усіма точками (не важливо ядрові чи ні) досяжними з неї. Кожен кластер містить принаймні одну ядрову точку. Не-ядрові точки можуть бути частиною кластера, тоді вони утворюють «ребро» кластера, оскільки з них не будуть досяжні інші точки.



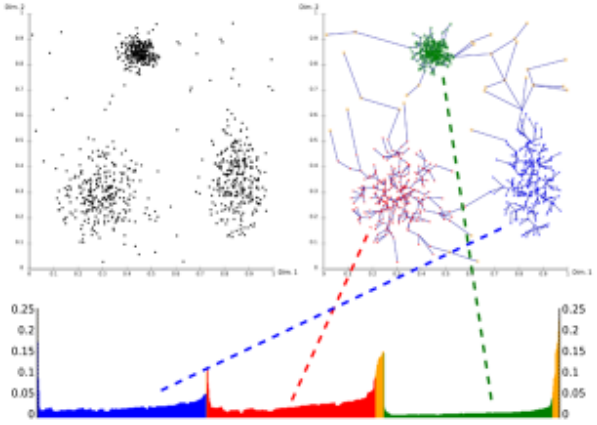
На зображенні minPts = 4. Точка A, як і червоні точки є ядровими, бо окіл цих точок радіуса ε містить щонайменше 4 точки, включно з нею самою. Оскільки, всі вони досяжні одна з одної, то вони утворюють спільний кластер.
Точки B та C не будуть ядровими, але досяжні з A або з якихось ядрових точок і тому належать кластеру. Точка N є шумовою, бо не буде ні ядровою ні безпосередньо досяжною.

Досяжність не є симетричним відношенням, бо, за визначенням, немає точок досяжних з не-ядрової точки (не-ядрова точка може бути досяжною, але відносно неї немає досяжних точок). Тому, поняття зв'язності потрібне для формального визначення степені кластеризації в DBSCAN. Дві точки p та q є щільно-зв'язними, якщо існує точка o з якої вони є досяжними. Щільна-зв'язність є симетричною.

Тоді кластер має дві властивості:

1. Всі точки кластера взаємно щільно-зв'язні.
2. Якщо точка є щільно-зв'язною зв'язною з будь-якою точкою кластера, то вона так само належить кластеру.

Метод **OPTICS** (*Ordering points to identify the clustering structure*)



Використовуючи *графік досяжності* (спеціальний вид дендрограми), легко отримати ієрархічну структуру кластерів. Це 2D графік, з упорядкуванням точок, оброблених OPTICS на осі Ox, а відстань *досяжності* на осі Oy. Так, як точки, що належать до одного кластера мають низьку *досяжність* для найближчих сусідів, кластери показують ділянки досяжності. Чим глибша ділянка, тим щільніший кластер.

Зображення вище ілюструє цю концепцію. У верхній лівій частині, показано штучний набір даних. Верхня права частина візуалізує спотучне дерево, створене за допомогою OPTICS, і нижня частина показує ділянку досяжності, обчислену OPTICS. Виявлення кластерів з допомогою цього графіку може бути зроблене вручну, вибравши діапазон на осі Ox після візуального огляду, вибравши поріг на осі Oy (результат буде схожий на результат DBSCAN кластеризації з тими ж ε і minPts параметри, тут значення 0,1 може дати добрі результати), або за допомогою різних алгоритмів, які намагаються виявити ділянки, використовуючи крутизну, виявлення коліна або локальні максимуми. Кластери, отримані таким чином можуть мати ієрархічну структуру, чого не може бути досягнуто за допомогою DBSCAN.

Метод **OPTICS** (*Ordering points to identify the clustering structure*)

Як і DBSCAN, OPTICS вимагає двох параметрів: ε, що є максимальною відстанню (радіусом) і $MinPts$, що показує мінімальну кількість об'єктів, необхідних для формування кластера. Точка(об'єкт) p є *ядром*, якщо хоча б $MinPts$ точок (об'єктів) знаходяться в межах ε-відстані $N_\varepsilon(p)$. Всупереч DBSCAN, OPTICS також розглядає об'єкти, які є частиною більш «щільних» кластерів, так, щоб кожна точка була присвоєна *основній відстані* до $MinPts$th її найближчої точки:

$$\text{core-dist}_{\varepsilon,MinPts}(p) = \begin{cases} \text{UNDEFINED} & \text{if } |N_\varepsilon(p)| < MinPts \\ MinPts\text{-th smallest distance to } N_\varepsilon(p) & \text{otherwise} \end{cases}$$

Досяжність до об'єкта o від об'єкта p знаходиться, як відстань між o і p, або, як *основна відстань* від p:

$$\text{reachability-dist}_{\varepsilon,MinPts}(o,p) = \begin{cases} \text{UNDEFINED} & \text{if } |N_\varepsilon(p)| < MinPts \\ \max(\text{core-dist}_{\varepsilon,MinPts}(p), \text{dist}(p,o)) & \text{otherwise} \end{cases}$$

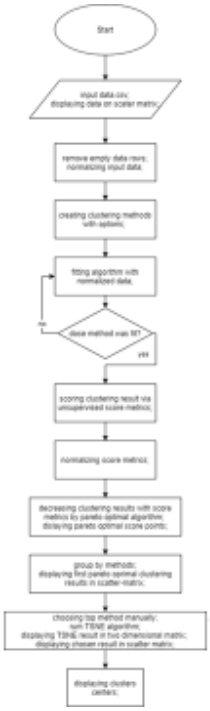
Якщо p і o є найближчими сусідами, тобто $\varepsilon' < \varepsilon$ ми повинні припускати, що p і o належать до одного кластеру.

Обидва значення *основної відстані* і *досяжності* є невідомими, якщо недоступний кластер достатньої щільності (w.r.t. ε). Якщо взяти велике значення ε, то цього не станеться, проте тоді кожен запит ε-сусідів може повертати навіть всю базу даних, що приведе до складності виконання $O(n^2)$. Отже, параметр ε потрібно вибрати, враховуючи щільність кластерів, які не представляють інтересу, щоб пришвидшити час виконання алгоритму.

Строго кажучи, параметр ε не є необхідним. Він можу бути просто встановленим максимально можливого значення.

Алгоритм

- Попередньо оцінити вхідні дані;
- Нормалізувати дані;
- Кластенрний аналіз;
- Візуалізація результату кластерного аналізу;
- Результати кластеризації - центри кластерів.



Критерії порівняння

- Індекс Calinski-Harabasz

Індекс Калінський-Харабаш набуває більшого (кращого) значення коли кластери щільні і чітко виділені[6]. Цей індекс ефективності кластеризації в більшій степені нагадує те, як людина оцінює результат кластеризації. Результат вважається хорошим, якщо візуально створені чітко виділені і щільно зосереджені групи міток. Індекс використовує два значеня:

W_k - дисперсія всередині кластера, матриця відстаней між елементами кластера та центром кластера.

B_k - дисперсія між групами, матриця відстаней між центром кластера і центрами всіх інших кластерів.

Для кожного елемента з набору даних E розмірності n, де в результаті роботи методу кластеризації маємо k кластерів, індекс Калінській-Харабаш рахується за формулою:

$$index = \frac{tr(B_k)}{tr(W_k)} * \frac{n-k}{k-1}$$

де $tr(B_k)$ - це слід між групами дисперсій матриці, а $r(W_k)$ - це слід між матрицею дисперсії внутрішніх груп кластеризації:

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T$$

$$B_k = \sum_{q=1}^{k} n_q (c_q - c_E)(c_q - c_E)^T$$

Критерії порівняння

- Індекс Davies-Bouldin

Краще значення - те що ближче до нуля. Індекс Девіс-Боулдіна означає середню схожість між кластерами, де мірою подібності є відстань між кластерами та розміри самих кластерів[5]. Як правило, вище (краще) значення набувається для округлих форм кластерів, ніж для інших видів. Наприклад для методів кластеризації, які основані на щільності розподілу (наприклад DBSCAN), оцінка буде гіршою.

Індекс схожості визначається між кожним з кластерів C_i, $i = 1, \dots, k$ та найбільш схожим C_j. В контексті цього індексу, схожість - це значення R_{ij} яке рахується за формулою:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

де s_i - середня відстань між кожною точкою кластеру i і центром цього кластеру (також відомого як «діаметр» кластеру); d_{ij} - відстань між центроїдами кластеру i та j.

Індекс Девіс-Боулдіна рахується за формулою:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij}$$

Критерії порівняння

- Коефіцієнт Silhouette

Коефіцієнт силуету - приклад такої оцінки, де вищий показник коефіцієнта силуету означає модель з більш чітко визначеними межами між кластерами. Коефіцієнт рахується за формулою (для кожної точки своє значення):

$$s = \frac{b - a}{\max(a, b)}$$

де a - середня відстань між точкою та всіма іншими точками цього кластеру; b - середня відстань між точкою та всіма точками наступного найближчого кластера.

Коефіцієнт набуває значення в інтервалі від -1 - поганий результат кластеризації, до 1 - для дуже щільних кластерів з чіткими межами. Значення 0 означає шо кластери склеюються дуже сильно.

Критерії порівняння

- Оптимальність по Парето.

Нехай X означає множину допустимих рішень деякої задачі, $x \in X$ - допустиме рішення. Припустімо, що кожне рішення $x \in X$ оцінюється по n критеріям $(n \geq 2)$. Мається дуже корисна конструкція для вирішення багатокритеріальної задачі - поняття оптимальності по Парето.

Нехай $H_i(x), x \in X$ - функція значення якої є оцінкою рішення $x \in X$ по критерію $i, i = \overline{1,n}$. Тоді вектор $H(x) = (H_1(x), \dots, H_l(x)), x \in X$ - набір оцінок рішення $x \in X$ по всім критеріям. Припустімо, що ступінь приналежності рішення $x \in X$ зростає разом з зростанням компонентів вектора H, тобто чим більше значення набуває $H_i(x)$, тим краще рішення x за критерієм $i, i = \overline{1,n}$.

Рішення $x^* \in X$ називається парето-оптимальним, якщо не існує іншого рішення $x \in X$, для якого

$$H_i(x) \geq H_i(x^*), i = \overline{1,n},$$
$$\exists i_0 : H_{i_0} > H_{i_0}(x^*).$$

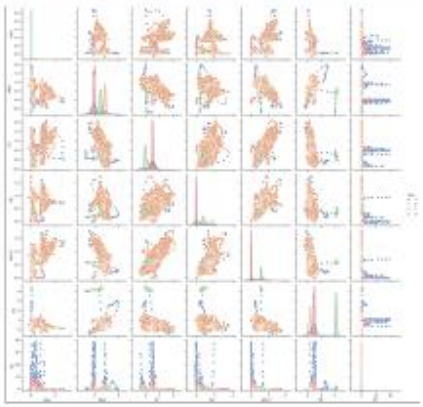
Іншими словами, якщо $x^* \in X$ - парето-оптимальне рішення, то не існує іншого рішення $x \in X$, яке є кращим рішення x^*, хоча б за одним із критеріїв, а за іншими критеріями не гірше[12].

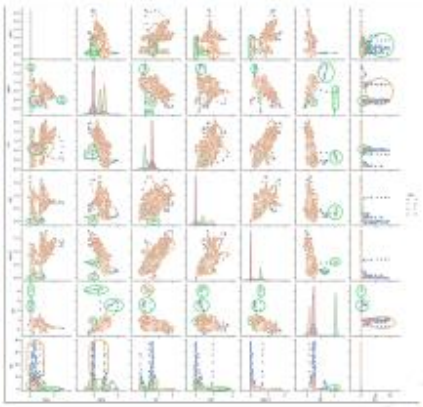
Для того щоб рішення $x \in X$, було оптимальним по парето , необхідно і достатньо щоб воно було рішенням наступної задачі багатокритеріальної оптимізації:

$$\max_{x \in X} \sum_{i=1}^{n} \lambda_i H_i(x),$$

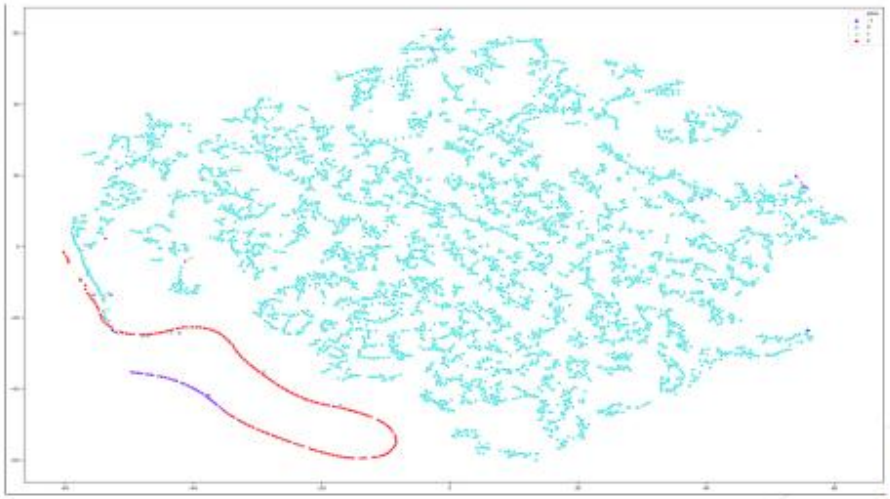
де параметри λ_i, $i = \overline{1,n}$, задовольняють умовам $\sum_{i=1}^{n} \lambda_i = 1$, $\lambda_i \in (0,1)$.

Аналіз отриманих результатів





Аналіз отриманих результатів



Аналіз отриманих результатів

	CALI	DEN	DT	GR	NEUT	PE	RT
-1	0.189984	2.235100	218.487549	57.781666	0.062674	4.972510	35490.766335
0	0.194080	2.602232	229.466117	141.515131	0.201233	3.299244	45.014042
1	0.162926	2.372065	192.933880	87.145291	0.182964	12.235160	289.092297
2	0.184291	2.067170	230.149204	23.194469	-0.027436	4.622646	4719.441183

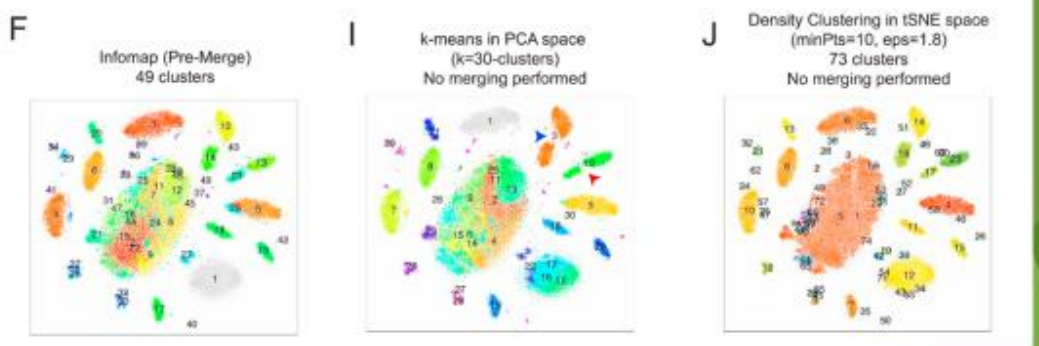
Подальша конвертація індексів в справжні назви корисних копалин, характеристики яких, з деякою похибкою (залежить від якості кластеризації, що в свою чергу дуже залежить від розміру і якості вхідного набору), співпадатимуть зі значеннями центрів кластерів унеможливлюється через те, що:

- невідомий хоча б регіон де проводились заміри;
- невідомий тип, виробник, та фізичні характеристики устаткування;
- невідомі налаштування сенсорів, серед яких є дуже чутливі (найменша відмінність від попередніх налаштувань яких – може призвести до кардинально інших результатів замірювань).

Висновки

- Сучасні методи кластеризації в області поверхневого аналізу геофізичних досліджень є дуже перспективними. Добре навчена модель може допомогти швидко проводити аналіз великих наборів даних, однакових за набором характеристик - параметрів.
- Аналізуючи цю роботу що найкращі результати кластеризації, для розглядуваних в цій роботі даних, демонструють методи алгоритм яких базується на щільнісній характеристиці вхідного набору, а також мають властивість добре опрацьовувати «шуми». Наприклад DBSCAN та його по суті модифікаця - OPTICS.
- Через те що актуальність даних існує лише в доволі короткому проміжку часу (на це впливають погодні умови) та через те що прилади і методи замірів геофізичних досліджень не є уніфікованою - швидкість навчання на актуальних даних є доволі вагомою характеристикою моделі, для поверхневого аналізу. Це дозволить не застосовувати додаткові ресурсу (в разі явної відсутності підземних ресурсів , за для здобутку яких і проводять дослідження), для більш детального і точного аналізу, тим самим зекономив гроші та час на ділянках землі, що не мають перспектив для здобутку шуканих корисних копалин.
- Недоліком цієї роботи являється те, що для фінальної ідентифікації кластерів як ту чи іншу корисну копалину - потрібна експертна, в цій галузі, людина що зможе провести ідентифікацію зважаючи на характеристику і налаштування інструменту, яким проводять підземний аналіз.

Шляхи для подальшого розвитку



Дякую за увагу!

ДОДАТОК Б ТЕКСТ ПРОГРАМИ

```
import warnings
warnings.filterwarnings('ignore')

from matplotlib import pyplot
from pandas import read_csv
from pandas.plotting import scatter_matrix
path = 'data/s_44.csv'
fields = [
   'MD',
   'CALI',
   'DEN',
   'DT',
   'GR',
   'NEUT',
   'PE',
   'RT',
]
input_data = read_csv(path, sep=';', skipinitialspace=True, usecols=fields, dtype=float)

import seaborn as sns
sns.set(style="ticks", color_codes=True)
g = sns.pairplot(input_data, palette="husl", diag_kind="kde")



clear_data_sets = {
   'Drop (without MD)': (input_data.dropna(), ['CALI','DEN','DT','GR','NEUT','PE','RT']),
   }
```

```
normalized_by_quantile_data_sets = {}
quantiles = ((0.05, 0.95), (0.1, 0.9))

for qt in quantiles:
    for name, (data, fields) in clear_data_sets.items():
        col_MD = data['MD']
        data_slice = data[fields]
        data = (data_slice - data_slice.quantile(qt[0])) / (data_slice.quantile(qt[1]) - data_slice.quantile(qt[0]))
        data['MD'] = col_MD
        normalized_by_quantile_data_sets[f'{name} norm by quantile ({qt[0]}-{qt[1]})'] = (data, fields)

RESULTS_DIR_PATH = 'normalized_by_quantile_result'

import warnings
warnings.filterwarnings('ignore')

from sklearn import cluster
import numpy as np
import pandas as pd
from tqdm import tqdm


for data_name, (data, fields) in normalized_by_quantile_data_sets.items():
    print(data_name)
    result_path_dir = RESULTS_DIR_PATH.strip('/')
    methods = [
        *[
            cluster.AffinityPropagation(damping=damping, convergence_iter=convergence_iter)
            for damping in np.arange(0.5, 1, 0.1)
            for convergence_iter in range(15, 55, 10)
```

```
            ],
        *[
            cluster.AgglomerativeClustering(n_clusters=n_clusters, linkage=linkage)
            for n_clusters in range(3, 13)
            for linkage in ['ward', 'complete', 'average', 'single']
            ],
        *[
            cluster.Birch(n_clusters=n_clusters, threshold=threshold,
branching_factor=branching_factor)
            for n_clusters in range(3, 13)
            for threshold in np.arange(0.1, 1.1, 0.1)
            for branching_factor in range(10, 101, 10)
            ],
        *[
            cluster.DBSCAN(min_samples=min_samples, eps=float(f"{eps:0.2f}"),
algorithm=algorithm, n_jobs=100)
            for min_samples in range(3, 11)
            for eps in np.arange(0.1, 1.1, 0.1)
            for algorithm in ['ball_tree', 'kd_tree', 'brute']
            ],
        *[
            cluster.KMeans(n_clusters=n_clusters, algorithm=algorithm, n_jobs=100)
            for n_clusters in range(3, 13)
            for algorithm in ['auto', 'full', 'elkan']
            ],
        *[
            cluster.MiniBatchKMeans(n_clusters=n_clusters, batch_size=batch_size,
reassignment_ratio=float(f"{reassignment_ratio:0.2f}"))
            for n_clusters in range(3, 13)
            for batch_size in range(50, 401, 50)
            for reassignment_ratio in np.arange(0.01, 1, 0.08)
            ],
        *[
```

```
                cluster.MeanShift(bandwidth=bandwidth, n_jobs=100)
                for bandwidth in range(1, 10)
                ],
            *[
                cluster.OPTICS(min_samples=min_samples, max_eps=max_eps, metric=metric, cluster_method=cluster_method, n_jobs=100)
                for min_samples in range(5, 21, 5)
                for max_eps in [1, 5, 10, 100 ,1000, np.inf]
                for metric in [
                  'cityblock', 'cosine', 'euclidean', 'l1', 'l2', 'manhattan', 'braycurtis', 'canberra', 'chebyshev',
                  'correlation', 'dice', 'hamming', 'jaccard', 'kulsinski', 'minkowski', 'rogerstanimoto',
                  'russellrao', 'sokalmichener', 'sokalsneath', 'sqeuclidean', 'yule'
                 ]
                for cluster_method in ['xi', 'dbscan']
                ],
            *[
                cluster.SpectralClustering(n_clusters=n_clusters, eigen_solver=eigen_solver, affinity=affinity, gamma=gamma, assign_labels=assign_labels, n_jobs=100)
                for n_clusters in range(3, 13)
                for eigen_solver in ['lobpcg', 'arpack']
                for affinity in ['nearest_neighbors']
                for gamma in np.arange(0.1, 1, 0.1)
                for assign_labels in ['kmeans', 'discretize']
                ],
            *[
                cluster.SpectralBiclustering(n_clusters=n_clusters, method=method, n_components=n_components, n_best=n_best, svd_method=svd_method, mini_batch=mini_batch, n_jobs=100)
                for n_clusters in range(3, 8)
                for method in ['bistochastic', 'scale']
                for n_components in range(3, 13)
```

```
                for n_best in range(1, 4)
                for svd_method in ['randomized']
                for mini_batch in [True, False]
                ],
            *[
                cluster.SpectralCoclustering(n_clusters=n_clusters, svd_method=svd_method,
n_svd_vecs=n_svd_vecs, mini_batch=mini_batch, n_jobs=100)
                for n_clusters in range(3, 13)
                for svd_method in ['randomized']
                for n_svd_vecs in range(2, 6)
                for mini_batch in [True, False]
                ],
            ]
        for method in tqdm(methods):
            if str(method) in list(data.columns):
                continue

            method.fit(data[fields])
            try:
                data[str(method)] = method.labels_.astype(str)
            except AttributeError:
                data[str(method)] = method.row_labels_.astype(str)
        data.to_csv(f'{result_path_dir}/{data_name}/labeled.csv', index=False)


    from sklearn import metrics
    from tqdm import tqdm
    from IPython.display import clear_output
    import re



    def get_method_from_str(method_str: str):
        return pd.Series([re.search(r'^(.+)\(', method_str).group(1),
dict(re.findall(r'(\w+)=(.*?)[,\)]', method_str))])
```

```
result = {}
for data_name, (data, fields) in normalized_by_quantile_data_sets.items():
    result_path_dir = RESULTS_DIR_PATH.strip('/')
    tmp_data = data[fields]
    tmp_labels = data.drop([*fields,'MD'], axis=1)
    tmp_result = {}
    method_names = tmp_labels.columns
    for method_name in tqdm(method_names):
        clear_output(wait=True)
        if 10 > tmp_labels[method_name].nunique() > 3 :
            tmp_result[method_name] = {
            'Calinski and Harabasz Score': metrics.calinski_harabasz_score(tmp_data, tmp_labels[method_name]),
            'Davies-Bouldin Score': metrics.davies_bouldin_score(tmp_data, tmp_labels[method_name]),
            'Silhouette Score': metrics.silhouette_score(tmp_data, tmp_labels[method_name]),
            }
        result_df = pd.DataFrame(tmp_result).T
        result_df.reset_index(inplace=True)
        result_df[['method', 'params']] = result_df['index'].apply(get_method_from_str)
        result_df.to_csv(f'{result_path_dir}/{data_name}/metrics.csv', index=False)
        result[data_name] = result_df

from IPython.display import display, HTML


main_data = normalized_by_quantile_data_sets.copy()
metrics_scores = result
for data_name, metrics_score in metrics_scores.items():

    idx = metrics_score.groupby('method')['Silhouette Score'].transform(max) ==
```

```
metrics_score['Silhouette Score']
        best_methods_results = metrics_score[idx].sort_values(by=['Silhouette Score'], ascending=False).drop_duplicates('method', keep='first')

        display(HTML(f'<h2>{data_name}</h2>'), best_methods_results)

        bufer = main_data[data_name]
        main_data[data_name] = {'data': bufer[0], 'fields':  bufer[1], 'best_labels': []}

        for val  in zip(best_methods_results['index'],best_methods_results['method']):
            main_data[data_name]['best_labels'].append(val)

    from IPython.display import display, HTML


    main_data = normalized_by_quantile_data_sets.copy()
    metrics_scores = result
    for data_name, metrics_score in metrics_scores.items():

        normalized_metrics = metrics_score.copy()
        data_slice = normalized_metrics[['Calinski and Harabasz Score', 'Davies-Bouldin Score', 'Silhouette Score']]
        normalized_metrics[['Calinski and Harabasz Score', 'Davies-Bouldin Score', 'Silhouette Score']] = (data_slice - data_slice.quantile(0)) / (data_slice.quantile(1) - data_slice.quantile(0))

        revers_mertrics = normalized_metrics.copy()
        revers_mertrics[['Calinski and Harabasz Score', 'Davies-Bouldin Score','Silhouette Score']] *= -1

        nondominated = pareto.eps_sort([list(revers_mertrics.itertuples(False))], [1, 2, 3])

        df_pareto = pd.DataFrame.from_records(nondominated, columns=list(revers_mertrics.columns.values))
```

```
        display(HTML(f'<h2>{data_name}</h2>'), df_pareto.drop_duplicates('method', keep='first'))

        metrics_score['pareto'] = 0
        metrics_score['pareto'][metrics_score['index'].isin(list(df_pareto['index']))] = 1
        sns.pairplot(metrics_score[['Calinski and Harabasz Score', 'Davies-Bouldin Score', 'Silhouette Score', 'pareto']], hue='pareto')

        bufer = main_data[data_name]
        main_data[data_name] = {'data': bufer[0], 'fields':  bufer[1], 'best_labels': []}

        for val in zip(df_pareto.drop_duplicates('method', keep='first')['index'],df_pareto.drop_duplicates('method', keep='first')['method']):
            main_data[data_name]['best_labels'].append(val)



    import seaborn as sns
    import matplotlib.pyplot as plt

    main_data['Drop (without MD) norm by quantile (0.1-0.9)']['data'][['MD', 'CALI','DEN','DT','GR','NEUT','PE','RT']].plot(x='MD',subplots=True, figsize=(20,20))

    plt.tight_layout()
    plt.show()

    #%%

    import seaborn as sns

    best_labels_idx
    for data_name, data in main_data.items():
        for label_idx, method_name in data['best_labels']:
```

```
        sns.pairplot(data['data'][[*data['fields'], label_idx]], hue=label_idx)

    best_method_index = df_pareto.iloc[0]['index']
    best_method_labels = main_data['Drop (without MD) norm by quantile (0.1-0.9)']['data'][[best_method_index]]

    best_method_labels.iloc[:, 0].unique()

    from sklearn.manifold import TSNE
    tsne = TSNE(n_iter=600,n_iter_without_progress=100,perplexity=27,random_state=30)
    transformed = tsne.fit_transform(clear_data_sets['Drop (without MD)'][0][['MD', 'CALI','DEN','DT','GR','NEUT','PE','RT']])
    transformed_no_md = tsne.fit_transform(clear_data_sets['Drop (without MD)'][0][['CALI','DEN','DT','GR','NEUT','PE','RT']])

    import seaborn as sns

    fig, axs = plt.subplots(ncols=2, figsize=(30, 20))
    sns.scatterplot(transformed[:,0], transformed[:,1], hue=best_method_labels.iloc[:, 0], palette='rainbow', ax=axs[0])
    sns.scatterplot(transformed_no_md[:,0], transformed_no_md[:,1], hue=best_method_labels.iloc[:, 0], palette='rainbow', ax=axs[1])

    import seaborn as sns
    sns.pairplot(main_data['Drop (without MD) norm by quantile (0.1-0.9)']['data'][['CALI','DEN','DT','GR','NEUT','PE','RT', best_method_index]], hue=best_method_index)

    from sklearn.neighbors.nearest_centroid import NearestCentroid

    for data_name, data in main_data.items():
        all_columns = data['data'].columns.to_list()
        optics_columns = [col for col in all_columns if 'OPTICS' in col]
```

```
    for label_idx in data['data'][optics_columns]:
        df = data['data'][data['fields']].copy()
        df['lable'] = data['data'][label_idx]
        if 10 > data['data'][label_idx].nunique() > 3:
            sns.pairplot(df, hue='lable')

for data_name, data in main_data.items():
    all_columns = data['data'].columns.to_list()
    optics_columns = [col for col in all_columns if 'OPTICS' in col]
    for label_idx in data['data'][optics_columns]:
        df = data['data'][data['fields']].copy()
        df['lable'] = data['data'][label_idx]
        if df['lable'].nunique() == 4:
            sns.pairplot(df, hue='lable')
            break

for data_name, data in main_data.items():
    all_columns = data['data'].columns.to_list()
    optics_columns = [col for col in all_columns if 'OPTICS' in col]
    for label_idx in data['data'][optics_columns]:
        df = data['data'][data['fields']].copy()
        df['lable'] = data['data'][label_idx]
        if df['lable'].nunique() == 4:

            X = clear_data_sets['Drop (without MD)'][0][['CALI','DEN','DT','GR','NEUT','PE','RT']]
            Y = df['lable']

            tsne                                                                    =
TSNE(n_iter=600,n_iter_without_progress=100,perplexity=27,random_state=30)
            transformed = tsne.fit_transform(X)

            fig, axs = plt.subplots(figsize=(30, 20))
            sns.scatterplot(transformed[:,0], transformed[:,1], hue=Y, palette='rainbow')
```

```
            clf = NearestCentroid()
            clf.fit(X, Y)
            display(pd.DataFrame(clf.centroids_, clf.classes_,
columns=['CALI','DEN','DT','GR','NEUT','PE','RT']))
            break

    #%%

    tsne = TSNE(n_iter=600,n_iter_without_progress=100,perplexity=27,random_state=30)
    transformed = tsne.fit_transform(clear_data_sets['Drop (without MD)'][0][['MD',
'CALI','DEN','DT','GR','NEUT','PE','RT']])
    transformed_no_md = tsne.fit_transform(clear_data_sets['Drop (without
MD)'][0][['CALI','DEN','DT','GR','NEUT','PE','RT']])

    #%%

    tsne = TSNE(n_iter=600,n_iter_without_progress=100,perplexity=27,random_state=30)
    transformed = tsne.fit_transform(clear_data_sets['Drop (without MD)'][0][['MD',
'CALI','DEN','DT','GR','NEUT','PE','RT']])
    transformed_no_md = tsne.fit_transform(clear_data_sets['Drop (without
MD)'][0][['CALI','DEN','DT','GR','NEUT','PE','RT']])

    #%%

    for data_name, data in main_data.items():
      all_columns = data['data'].columns.to_list()
      optics_columns = [col for col in all_columns if 'OPTICS' in col]
      for label_idx in data['data'][optics_columns]:
        df = data['data'][data['fields']].copy()
        df['lable'] = data['data'][label_idx]
        if df['lable'].nunique() == 4:
```

```
            X = clear_data_sets['Drop (without MD)'][0][['CALI','DEN','DT','GR','NEUT','PE','RT']]
            Y = df['lable']

            tsne = TSNE(n_iter=600,n_iter_without_progress=100,perplexity=27,random_state=30)
            transformed = tsne.fit_transform(X)

            fig, axs = plt.subplots(figsize=(30, 20))
            sns.scatterplot(transformed[:,0], transformed[:,1], hue=Y, palette='rainbow')

            clf = NearestCentroid()
            clf.fit(X, Y)
            display(pd.DataFrame(clf.centroids_, clf.classes_, columns=['CALI','DEN','DT','GR','NEUT','PE','RT']))
        print(clf.predict([[0.415532, -0.688320, 0.522490, -0.272774,- 0.636609, 1.378210, 1.379470]]))

            break


    tsne = TSNE(n_iter=600,n_iter_without_progress=100,perplexity=27,random_state=30)
    transformed = tsne.fit_transform(clear_data_sets['Drop (without MD)'][0][['MD', 'CALI','DEN','DT','GR','NEUT','PE','RT']])
    transformed_no_md = tsne.fit_transform(clear_data_sets['Drop (without MD)'][0][['CALI','DEN','DT','GR','NEUT','PE','RT']])
```